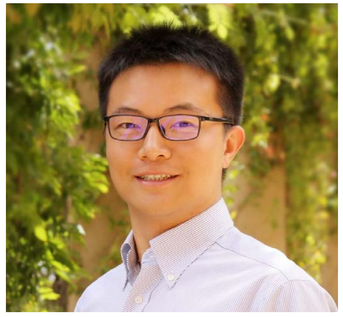
MIT HAN LAB

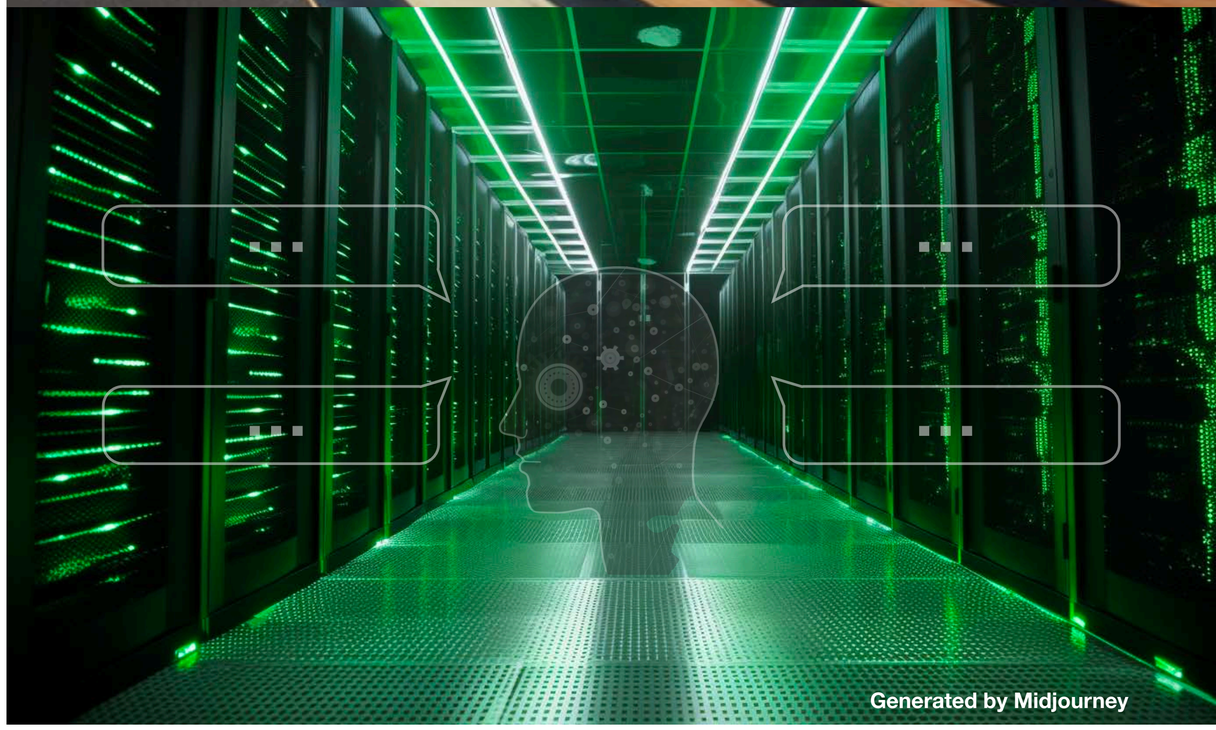
Hardware, AI, and Neural-nets
open source, co-design
http://github.com/mit-han-lab

Model Compression for Efficient AI Computing

From TinyML to LargeML

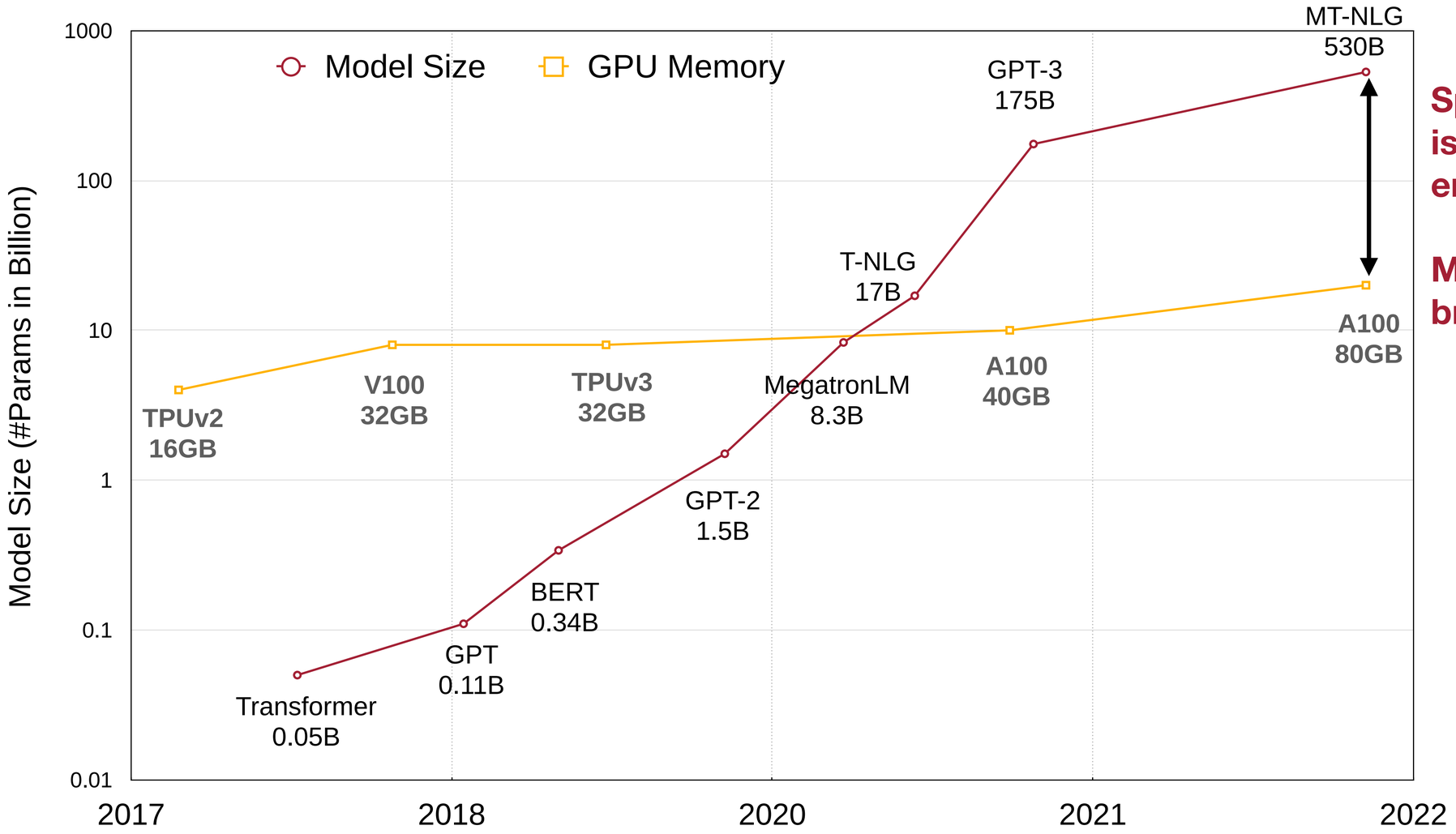


Song Han
MIT
songhan.mit.edu
tinyml.mit.edu
@SongHan_MIT



Model Compression

Bridges the Gap between the Supply and Demand of Computation



Specialized hardware is important but not enough.

Model compression bridges the gap.

Model Compression

Bridges the Gap between the Supply and Demand of Computation

AI Application
(demand of computation)

New Primitive → Hardware-aware NAS → Fine-Grained Pruning / CoarseGrained / Structured Pruning → Post-Training Quantization / Quantization-Aware Training → Distillation / Augmentation

AI Hardware
(supply of computation)

Model Compression

Applications

TR please briefly explain large language model in one sentence.

A large language model is a type of artificial intelligence that can process and generate human-like language, based on vast amounts of data it has been trained on.

Large Language Model

Generative AI

Advanced Driver Assistance System

TinyML

Techniques

Hardware-aware NAS

Pruning & Sparsity

Quantization

Distillation

New Primitive

Same Principle, Diverse Applications

Applications

TR please briefly explain large language model in one sentence.

A large language model is a type of artificial intelligence that can process and generate human-like language, based on vast amounts of data it has been trained on.

Large Language Model

Generative AI

Advanced Driver Assistance System

TinyML

Techniques

- Hardware-aware NAS
- Pruning & Sparsity
- Quantization
- Distillation
- New Primitive

Efficient Large Language Models

Reducing LLM Serving Cost and Accelerating Inference

We're experiencing exceptionally high demand. Please hang tight as we work on scaling our systems. ×

ChatGPT is at capacity right now

Get notified when we're back

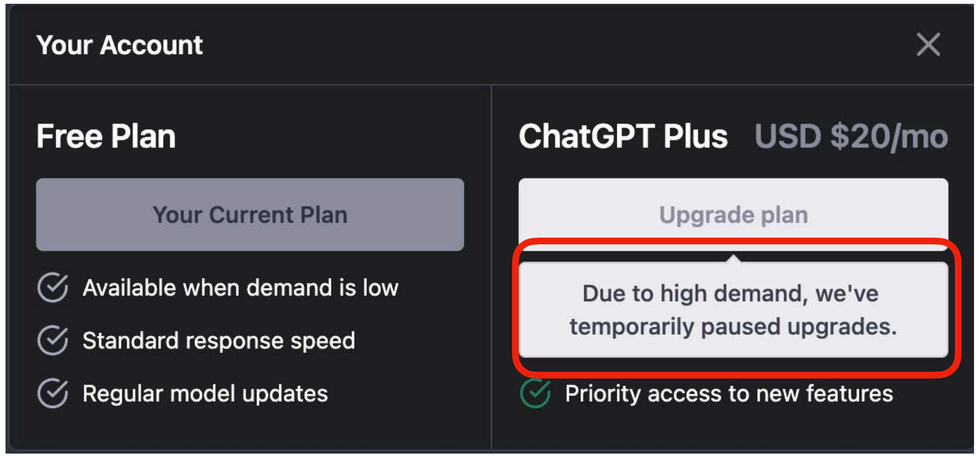
Write a rap about the status of ChatGPT.

Yeah, yo, what's up ChatGPT fam
A lot of people trying to jam
But don't worry, we got your back
Just check back soon, we'll get on track

ChatGPT's the place to be
For all your AI chat needs
We're working hard to keep up the pace
So hold tight, we'll be back in this space

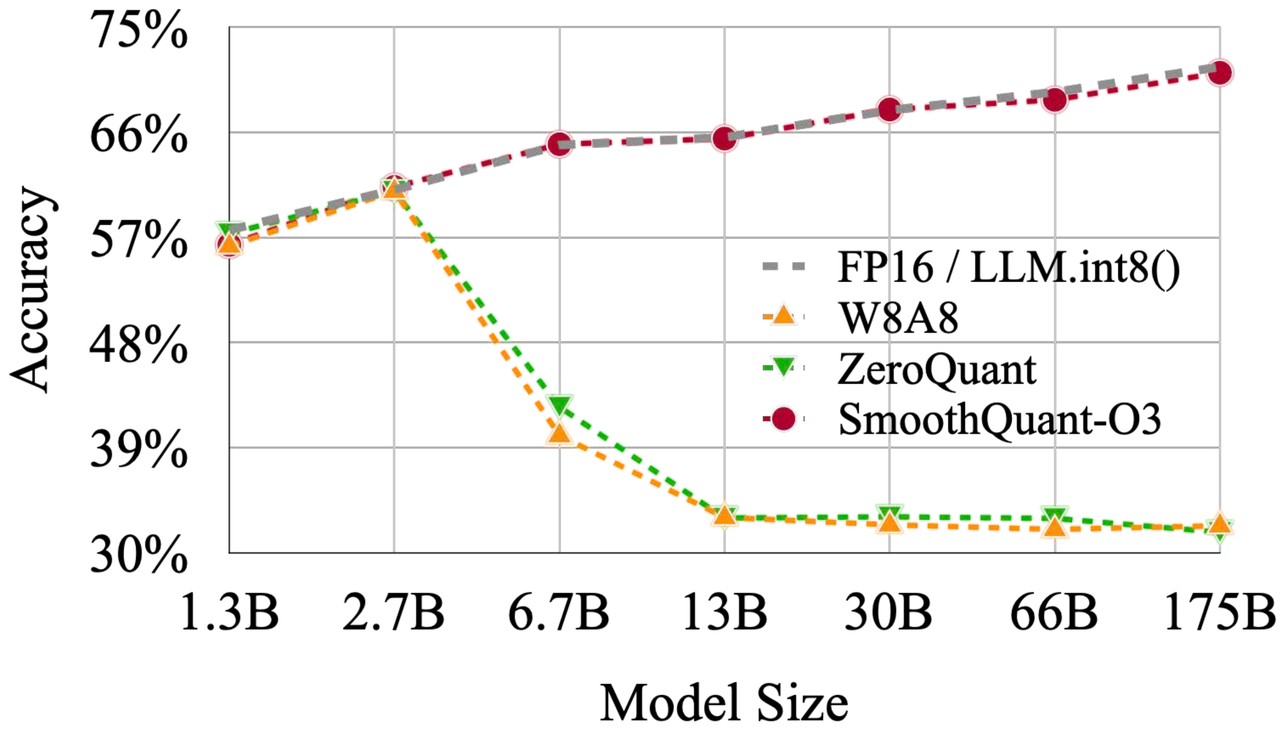
So don't you worry, don't be mad
We're working hard, it's not so bad
Just give us time, we'll be back
ChatGPT, the future of chat, that's a fact.

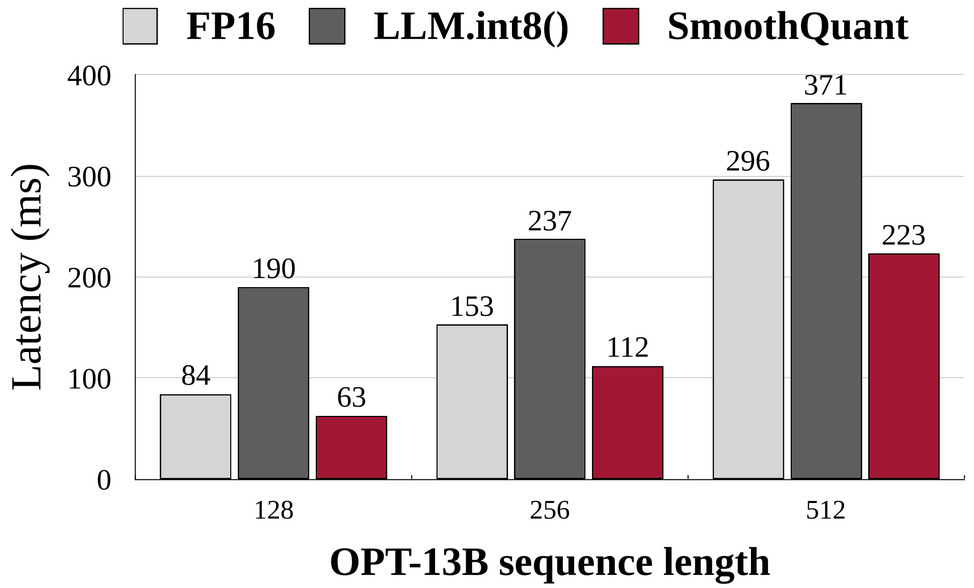
The LLM serving costs are extremely high



Quantization cut the model size by half, but…

Existing Quantization Method is Slow or Inaccurate



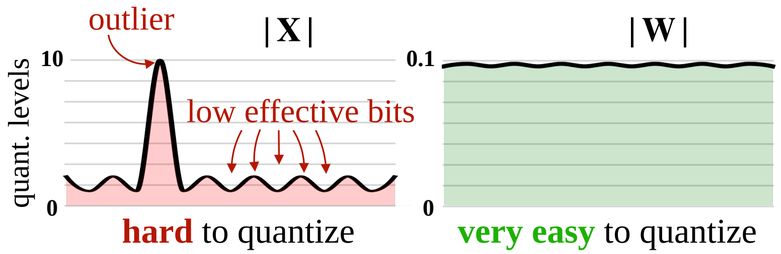


- Systematic outliers emerge in **activations** when we scale up LLMs beyond 6.7B. Traditional CNN quantization methods will destroy the accuracy.
- The accuracy-preserving baseline, LLM.int8() uses FP16 to represent outliers, which needs runtime outlier detection, scattering and gathering. It is slower than FP16 inference.

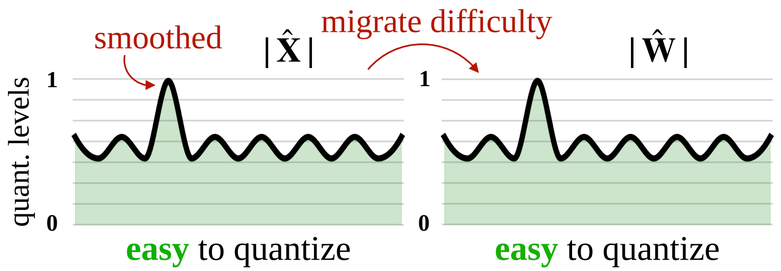
LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale (Dettmers *et al.*, 2022)

SmoothQuant

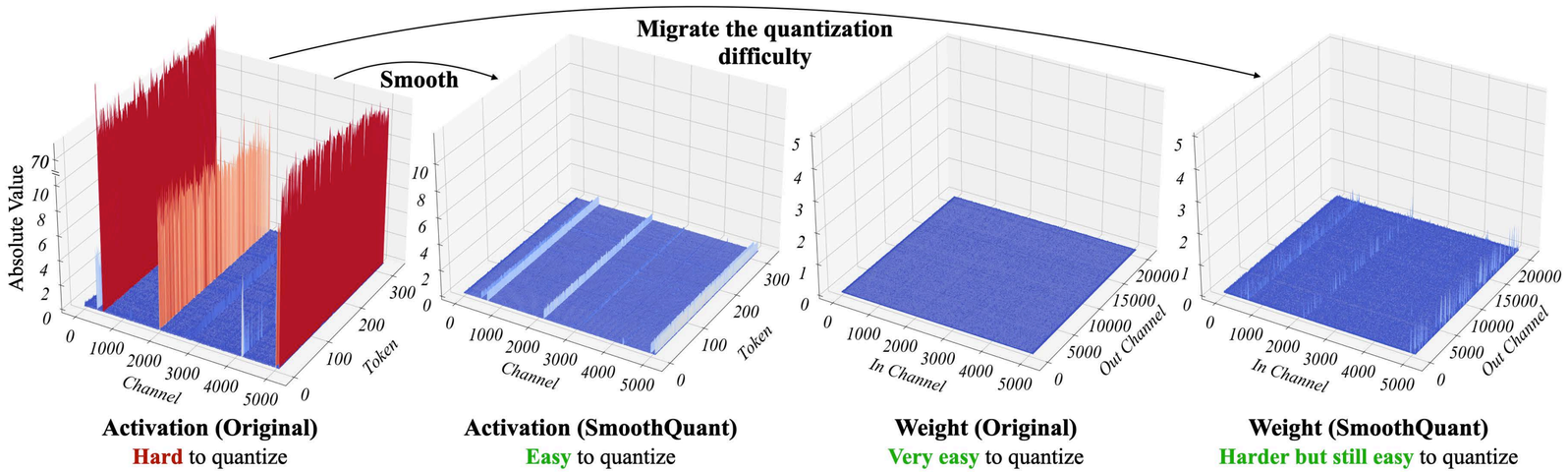
Lots of outliers; Migrate Quantization Difficulty from Activations to Weights



(a) Original



(b) SmoothQuant

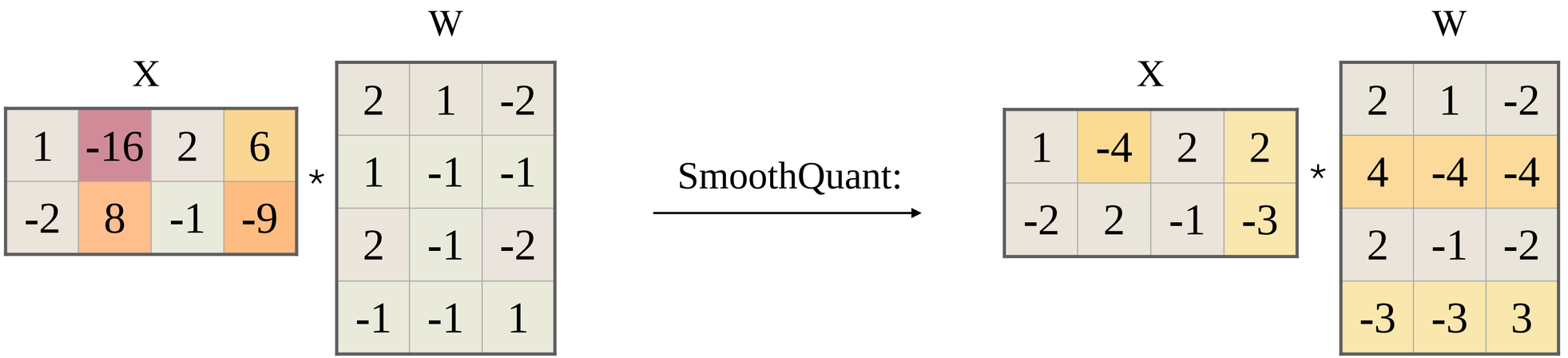


- Since matrix multiplication, A*B=C, is linear, we can shift information in A or B around. As such, we can balance the quantization difficulty across A and B.
- Since **weights are easy** to quantize while **activations are not**, SmoothQuant smooths the activation outliers by **migrating the quantization difficulty from activations to weights** with a mathematically equivalent transformation.

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models (Xiao et al., 2022)

SmoothQuant

Training-free, Offline Activation Smoothing



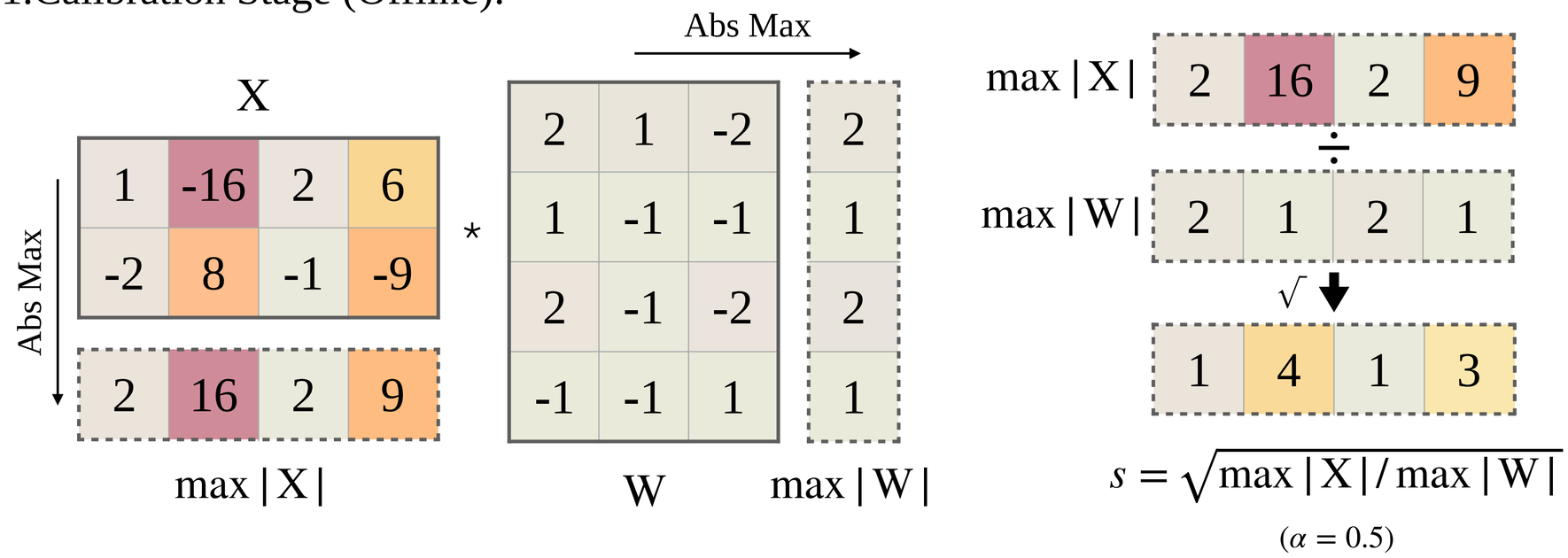
- Since matrix multiplication, A*B=C, is linear, we can shift information in A or B around.
- SmoothQuant smooths the activation outliers by **migrating the quantization difficulty from activations to weights** with a mathematically equivalent transformation.

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models (Xiao et al., 2022)

SmoothQuant

Training-free, Offline Activation Smoothing

1.Calibration Stage (Offline):



$$\mathbf{s}_j = \max(|\mathbf{X}_j|)^{\alpha} / \max(|\mathbf{W}_j|)^{1-\alpha},\ j = 1,2,\ldots,C_i$$

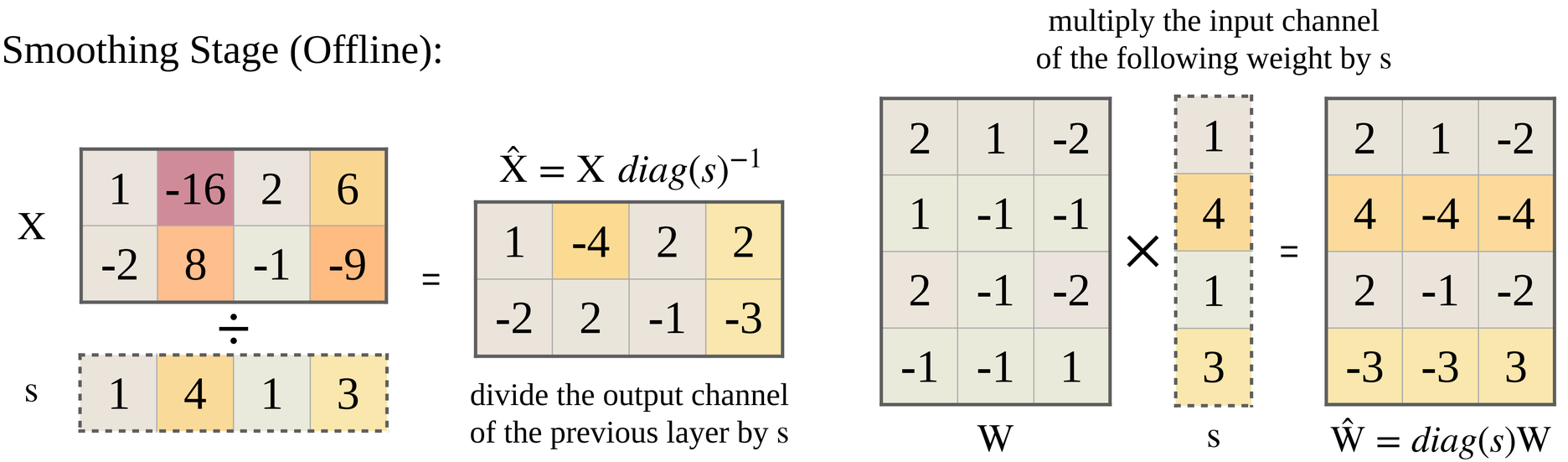
α: Migration Strength

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models (Xiao et al., 2022)

SmoothQuant

Training-free, Offline Activation Smoothing

2. Smoothing Stage (Offline):



$$\mathbf{s}_j = \max(|\mathbf{X}_j|)^{\alpha} / \max(|\mathbf{W}_j|)^{1-\alpha},\ j = 1,2,\ldots,C_i$$

$$\mathbf{Y} = (\mathbf{X}diag(\mathbf{s})^{-1}) \cdot (diag(\mathbf{s})\mathbf{W}) = \hat{\mathbf{X}}\hat{\mathbf{W}}$$

α: Migration Strength

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models (Xiao et al., 2022)

SmoothQuant

Training-free, Offline Activation Smoothing

3. Inference (deployed model):

$\hat{X}$

1	-4	2	2
-2	2	-1	-3

At runtime, the activations are smooth and easy to quantize

*

2	1	-2
4	-4	-4
2	-1	-2
-3	-3	3

$\hat{W}$

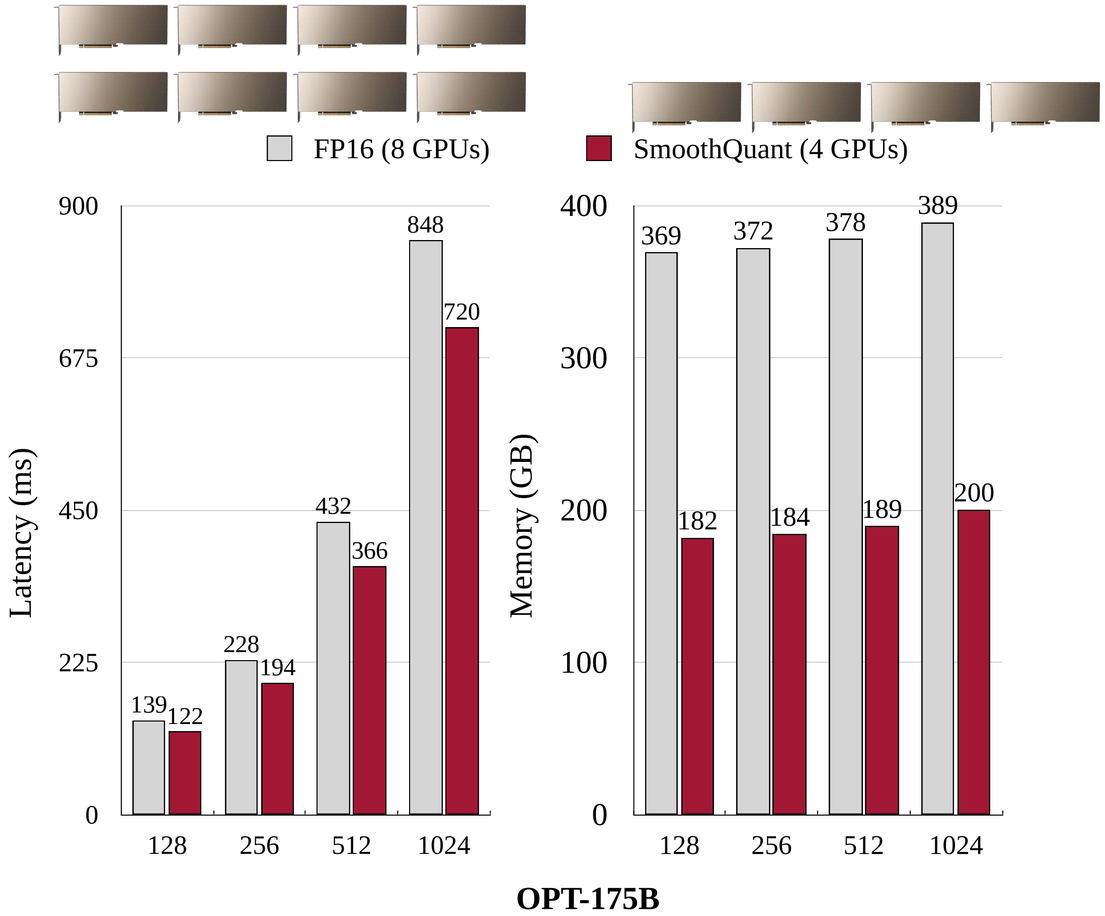
$$\mathbf{Y} = \hat{\mathbf{X}}\hat{\mathbf{W}}$$

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models (Xiao et al., 2022)

SmoothQuant

SmoothQuant is Accurate and Efficient

Method	OPT-175B	BLOOM-176B	GLM-130B*
FP16	71.6%	68.2%	73.8%
W8A8	32.3%	64.2%	26.9%
ZeroQuant	31.7%	67.4%	26.7%
`LLM.int8()`	71.4%	68.0%	73.8%
Outlier Suppression	31.7%	54.1%	63.5%
SmoothQuant-O1	**71.2%**	68.3%	**73.7%**
SmoothQuant-O2	71.1%	**68.4%**	72.5%
SmoothQuant-O3	71.1%	67.4%	72.8%



- SmoothQuant well maintains the accuracy without finetuning.
- SmoothQuant can both accelerate inference and halve the memory footprint.

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models (Xiao et al., 2022)

SmoothQuant

Enabling Serving the MT-NLG 530B Model within a Single Node

MT-NLG 530B Accuracy

	LAMBADA	HellaSwag	PIQA	WinoGrande	Average
FP16	76.6%	62.1%	81.0%	72.9%	73.1%
INT8	77.2%	60.4%	80.7%	74.1%	73.1%

MT-NLG 530B Efficiency

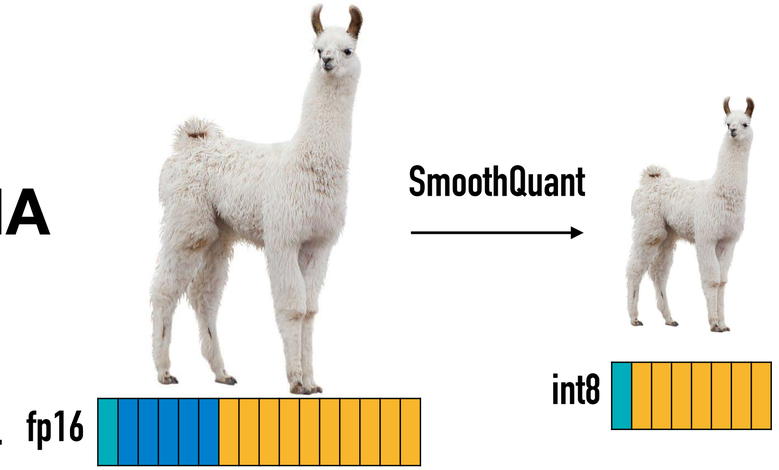
SeqLen	Prec.	#GPUs	Latency	Memory
512	FP16	16	838ms	1068GB
	INT8	8	839ms	545GB
1024	FP16	16	1707ms	1095GB
	INT8	8	1689ms	570GB

SmoothQuant can accurately quantize MT-NLG 530B model and reduce the serving GPU numbers by half at a similar latency, which allows serving the 530B model within a single node.

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models (Xiao et al., 2022)

SmoothQuant

Advancing new efficient open model LLaMA



- **LLaMA** (and its successors like Alpaca) are popular open-source LLMs, which introduced SwishGLU, making activation quantization even harder
- SmoothQuant can losslessly quantize LLaMA families, further lowering the hardware barrier

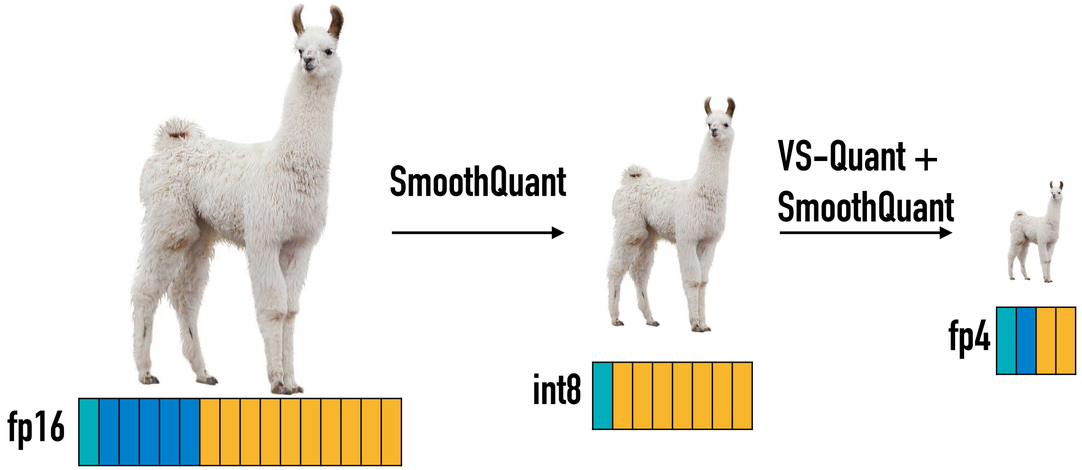
PIQA↑	LLaMA 7B	LLaMA 13B	LLaMA 30B	LLaMA 65B
FP16	78.24%	79.05%	80.96%	81.72%
SmoothQuant	78.24%	78.84%	80.74%	81.50%

Wikitext↓	LLaMA 7B	LLaMA 13B	LLaMA 30B	LLaMA 65B
FP16	11.51	10.05	7.53	6.17
SmoothQuant	11.69	10.31	7.71	6.68

W8A8 per token

SmoothQuant

Going smaller: W4A4 (FP4)

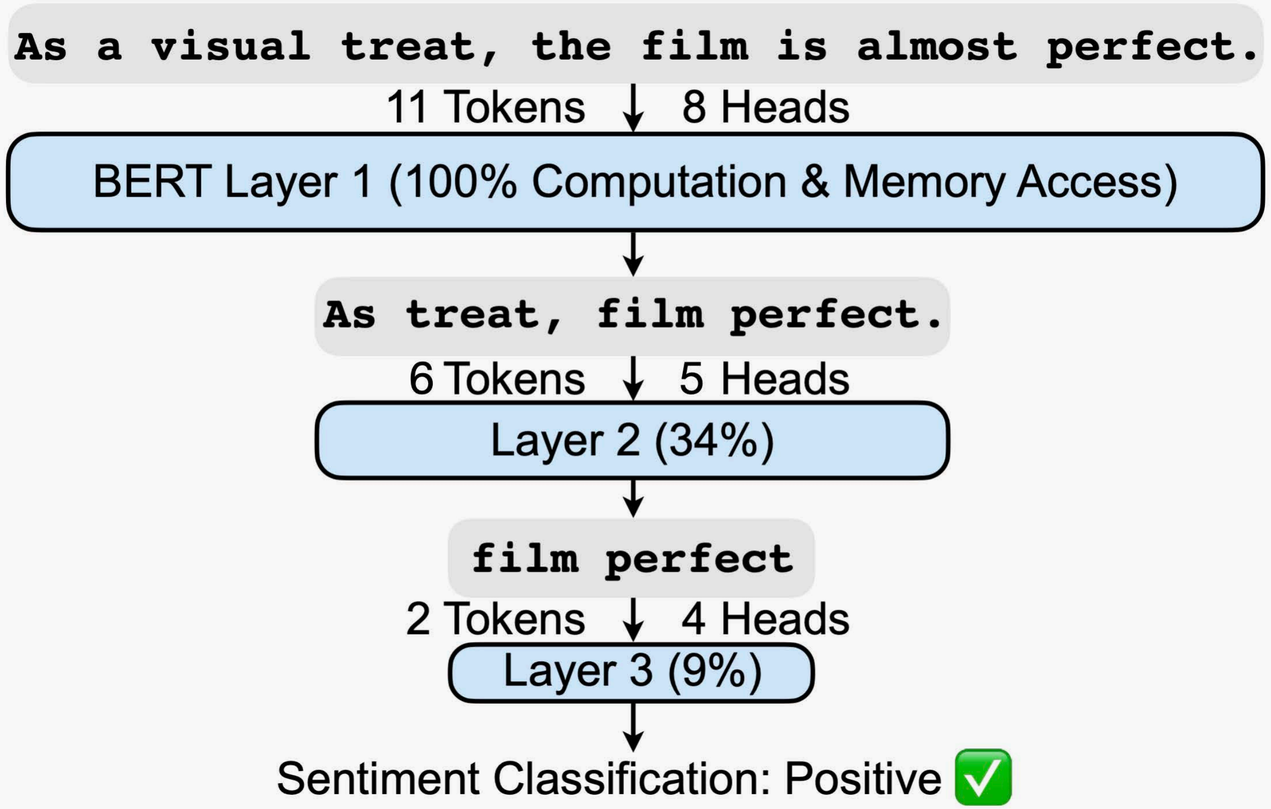


- Can we further push the frontier?
- We evaluate the W4A4 quantization for future-generation of hardware
  - Setting: FP4 data type with a block size of 64
- Red: ppl degrade > 0.5, Green: ppl degrade < 0.5. SmoothQuant helps most of the time.

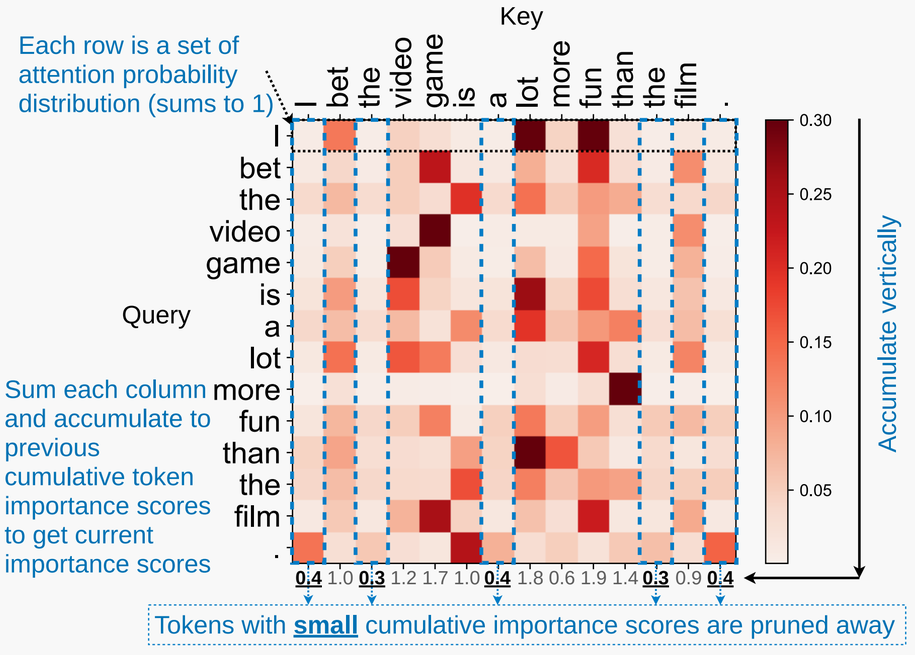
wikitext2	llama-7b	llama-30b	llama-65b
fp16	9.49	6.91	4.96
w4a4-m1e2-g64	10.2676	8.1453	5.4746
w4a4-m1e2-g64-sm0.25	10.1437	7.0089	5.4336
	opt-6.7b	opt-13b	opt-30b
fp16	15.12	14.13	13.09
w4a4-m1e2-g64	16.2289	14.7355	13.6172
w4a4-m1e2-g64-sm0.25	15.5899	14.5469	13.3931

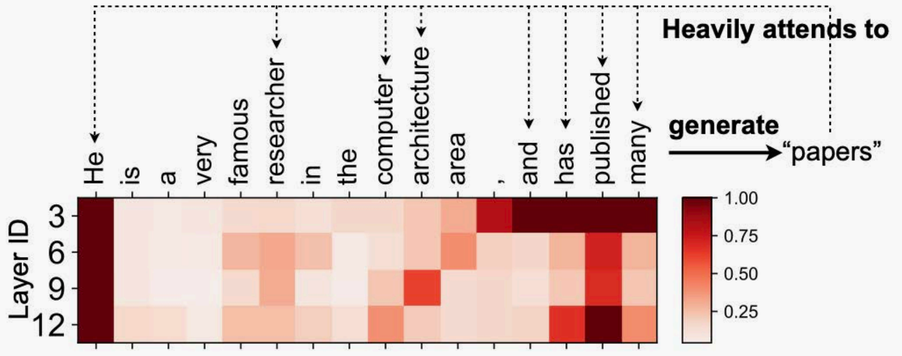
SpAtten: Transformer with Sparse Attention

Token Pruning: not every token are created equal



Remove redundant token and head according to cumulative importance



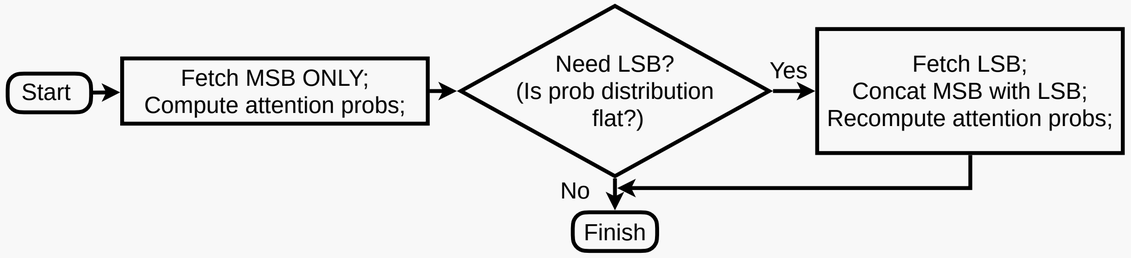


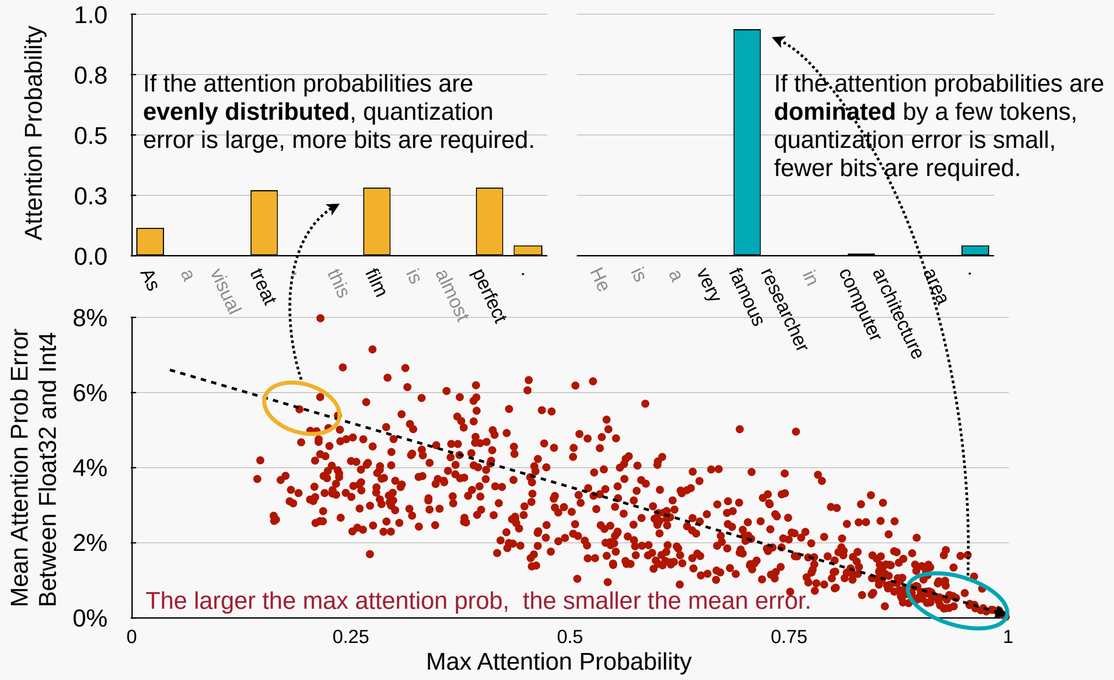
Cumulative importance scores in GPT-2. Unimportant tokens are pruned on the fly. Important tokens are heavily attended.

SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning [HPCA'21]

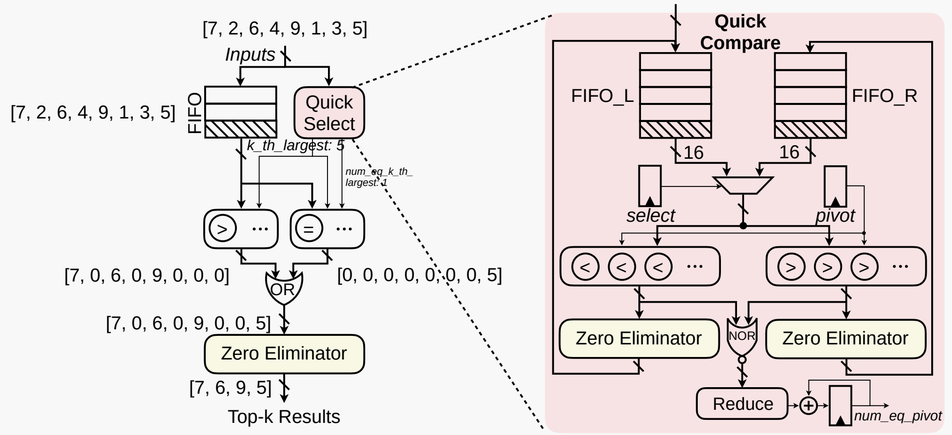
SpAtten: Transformer with Sparse Attention

Progressive Quantization: high confident attention requires low bit width

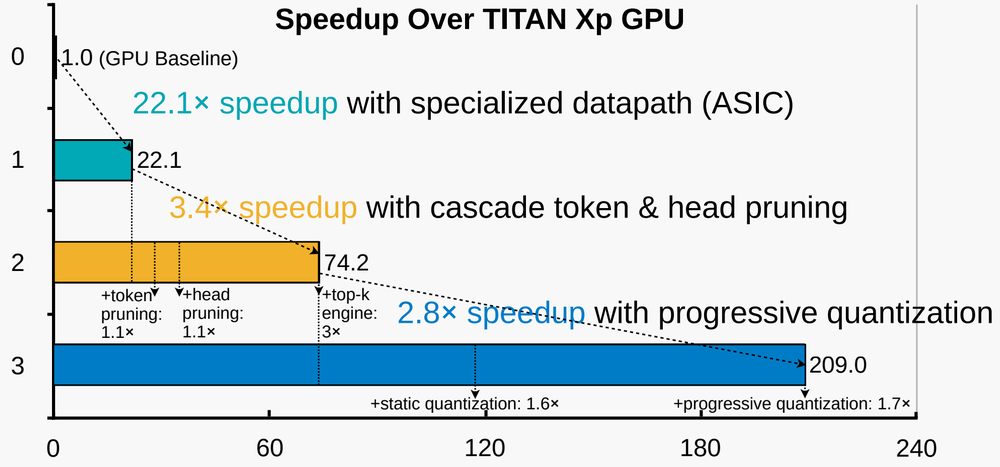




Progressive Quantization: only use MSB when attention confidence is high



Specialized top-k engine to select important token and head

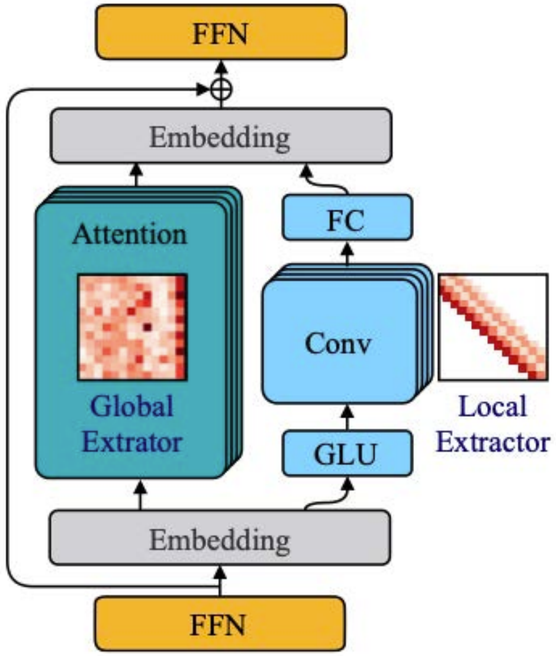


SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning [HPCA'21]

Lite Transformer

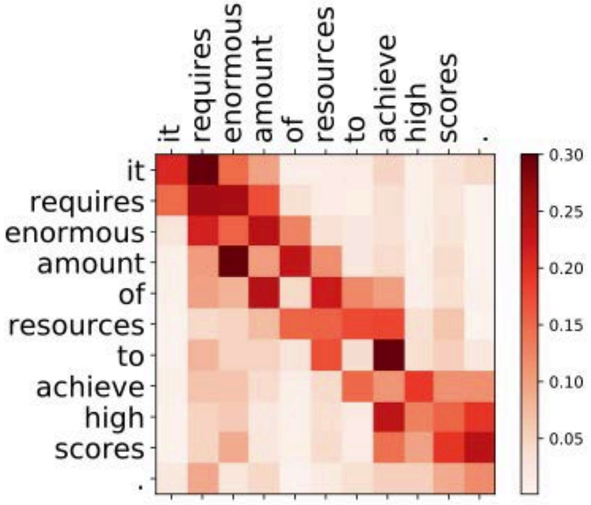
Local Convolution + Global Attention

- Long-Short Range Attention (LSRA):
  - **Convolution**: Efficiently extract the **local** (short-range) features.
  - **Attention**: Tailored for **global** (long-range) feature extraction.



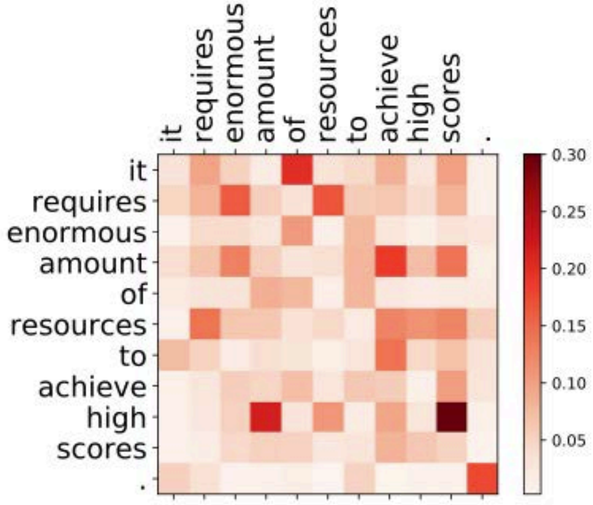
Original Attention

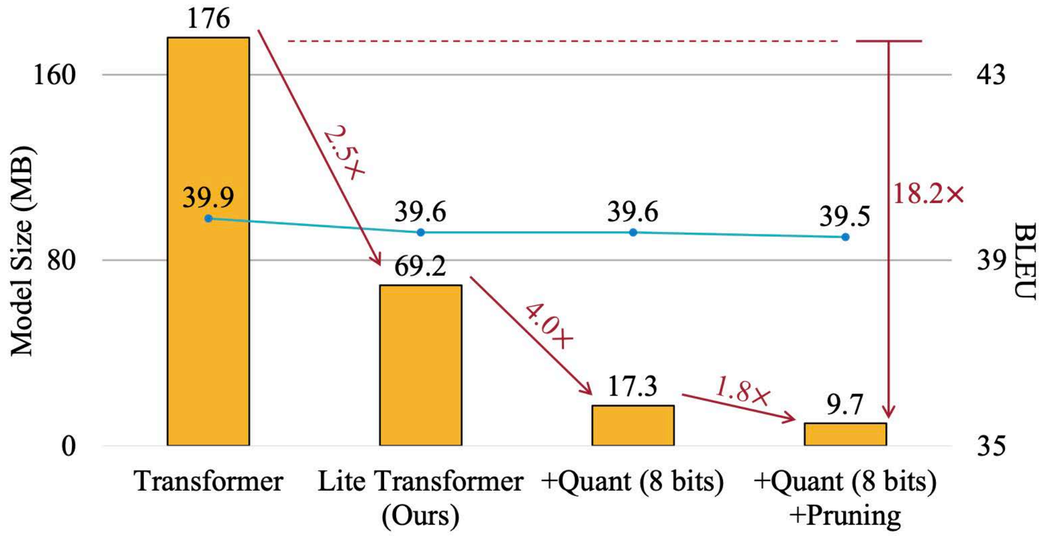
(Too much emphasize on local feature extraction)



Attention in LSRA

(Dedicated for global feature extraction)





Lite Transformer with Long-Short Range Attention [ICLR 2020]

Same Principle, Diverse Applications

Applications

Large Language Model

Generative AI

Advanced Driver Assistance System

TinyML

Techniques

Hardware-aware NAS

Pruning & Sparsity

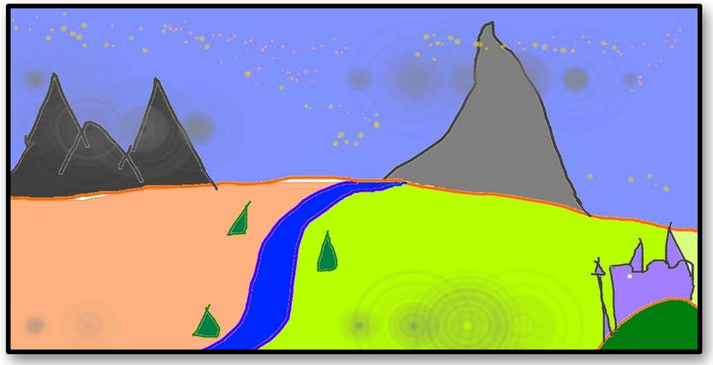
Quantization

Distillation

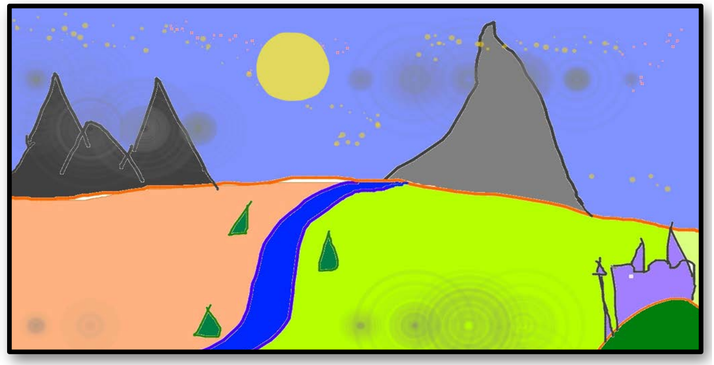
New Primitive

Compressing and Accelerating Diffusion Models

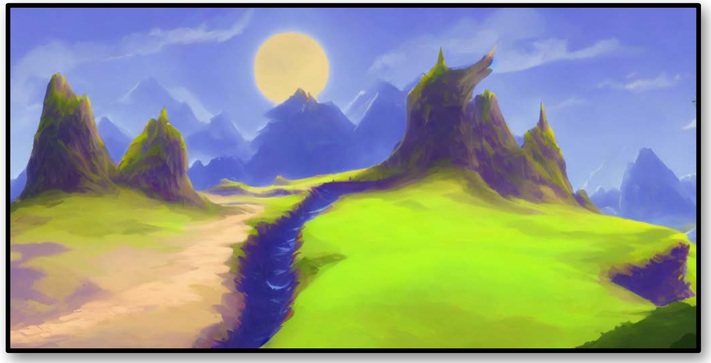
Prompt: A fantasy landscape, trending on artstation.



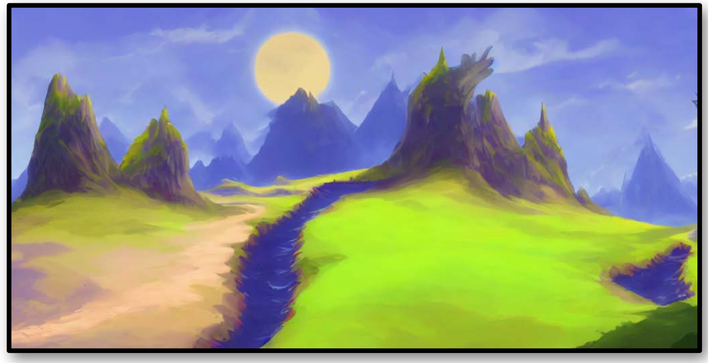
Original



Edited



Stable Diffusion+SDEdit:
1855GMACs, 369ms

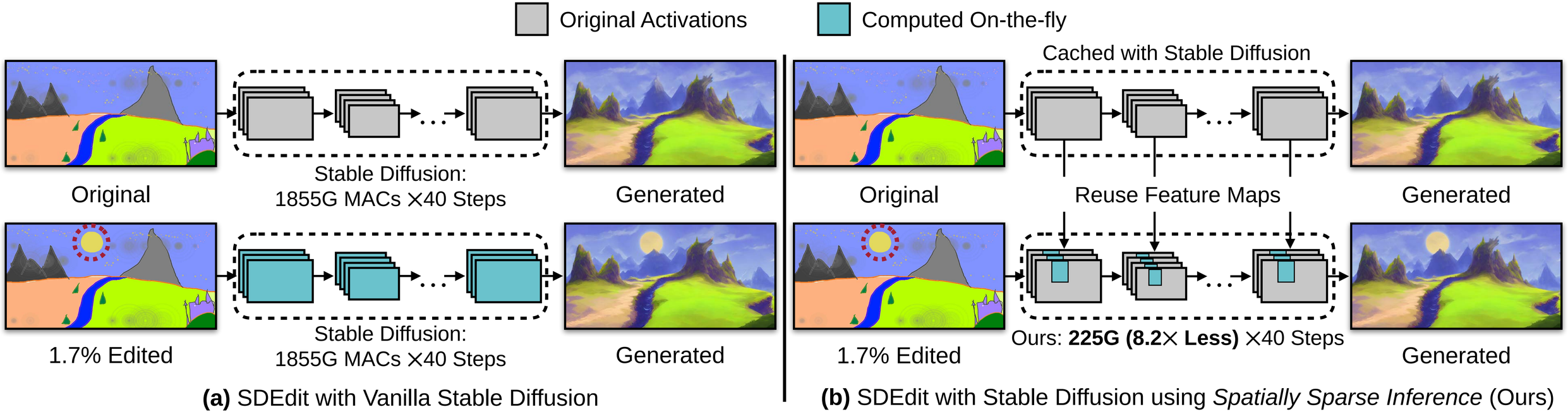


Ours:
225GMACs (8.2×),
51.2ms (7.2x)

Efficient Spatially Sparse Inference for Conditional GANs and Diffusion Models, NeurIPS'22

Spatially Sparse Inference for Diffusion Models

Vanilla Model Wastes Many Computations to Re-synthesize the Entire Image



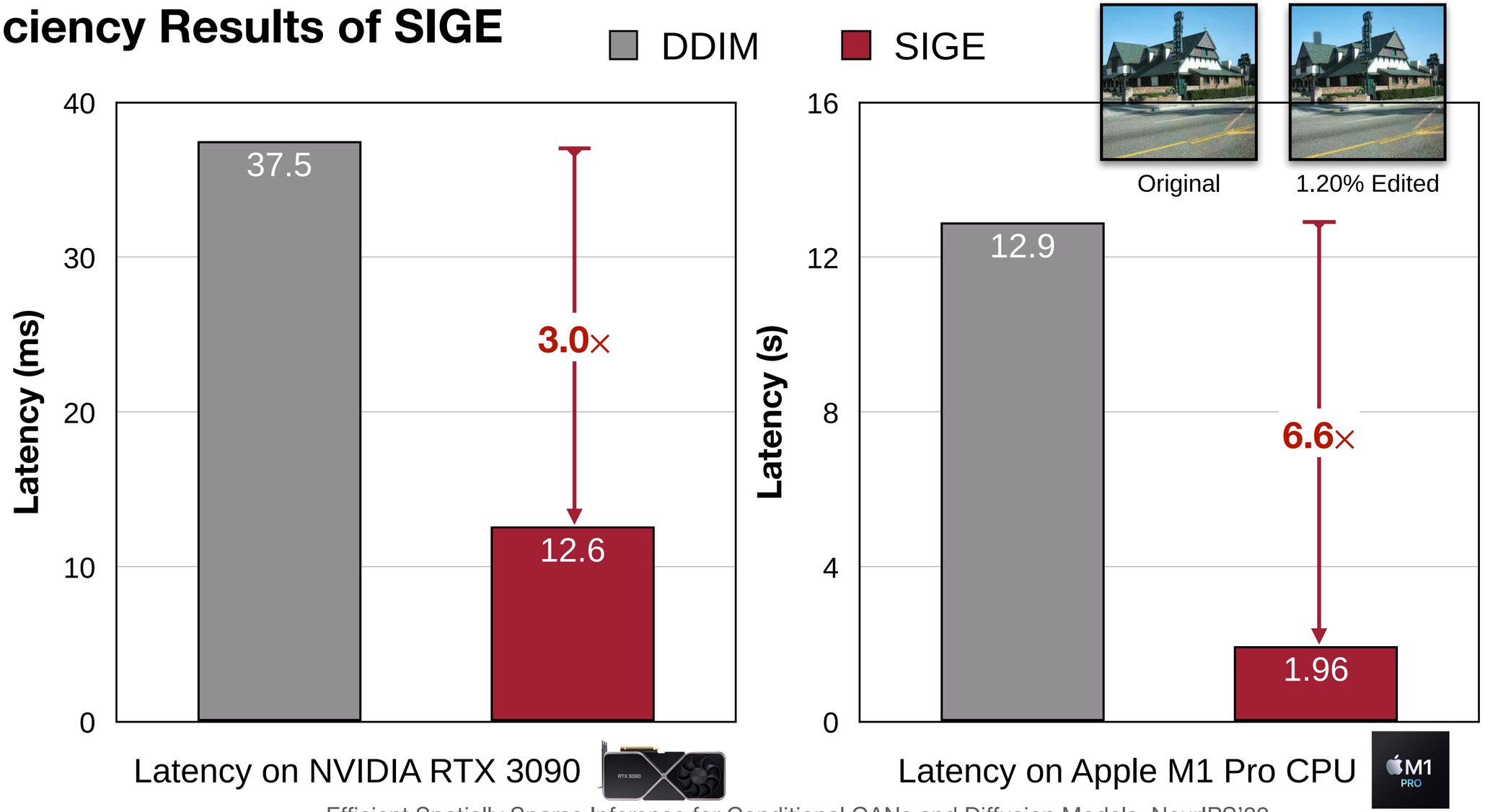
(a) SDEdit with Vanilla Stable Diffusion

(b) SDEdit with Stable Diffusion using *Spatially Sparse Inference* (Ours)

- Only 1.7% region is edited, but vanilla model re-synthesizes the entire image.
- Feature maps remain mostly the same at unedited regions.
- Reuse cached activations to selectively update edited regions (8× less computation).

Spatially Sparse Inference for Diffusion Models

Efficiency Results of SIGE



Efficient Spatially Sparse Inference for Conditional GANs and Diffusion Models, NeurIPS'22

Spatially Sparse Inference for Diffusion Models

Qualitative Results of SIGE on Stable Diffusion

A photograph of a horse on a grassland.

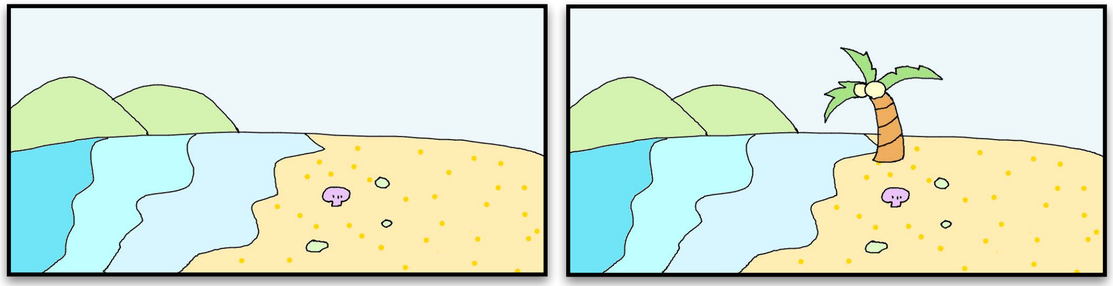
Original | 11.6% Masked

Stable Diffusion: 1855GMACs 369ms

Ours: 514G (3.6×) 95.0ms (3.9×)

Image Inpainting

A fantasy beach landscape, trending on artstation.



Original | 2.9% Edited

Stable Diffusion+SDEdit: 1855GMACs 369ms

Ours: 353G (5.3×) 76.4ms (4.8×)

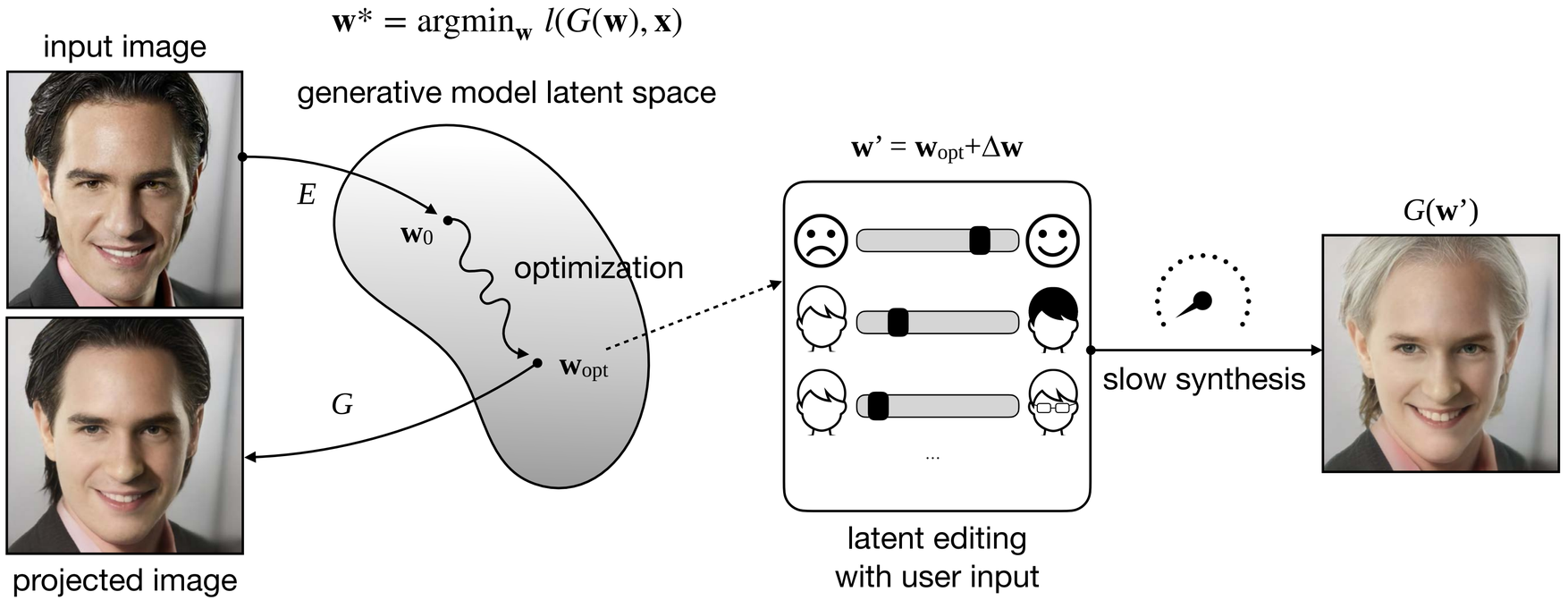
Image Editing

Latency Measured on NVIDIA RTX 3090

Anycost GAN

Image Editing With GANs

- We can edit an image by projecting an input image into the latent space of a GAN model, and then modify the latent code to generate a new image.
- Usually an **interactive** process, where users make **frequent** interactions.
- **Problem**: GAN model inference is **slow**; it usually requires a few seconds to get edited results, which hinders interactive use cases

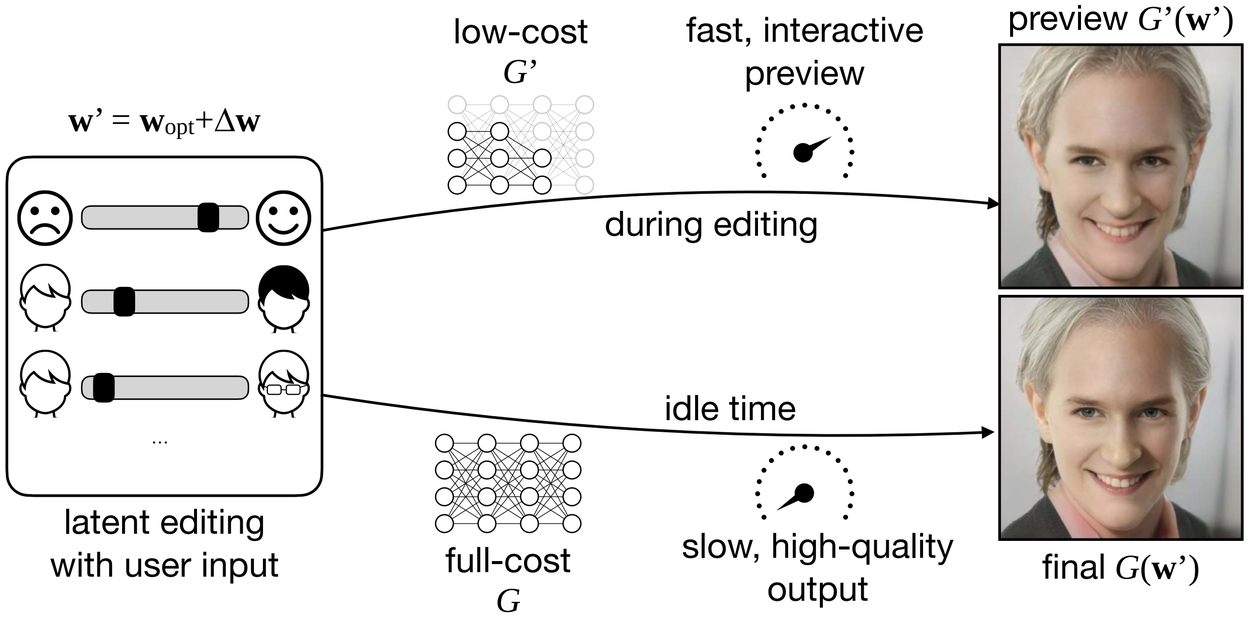


Anycost GANs for Interactive Image Synthesis and Editing [CVPR 2021]

Anycost GAN

Quick Preview By Runner A Smaller But Consistent Model

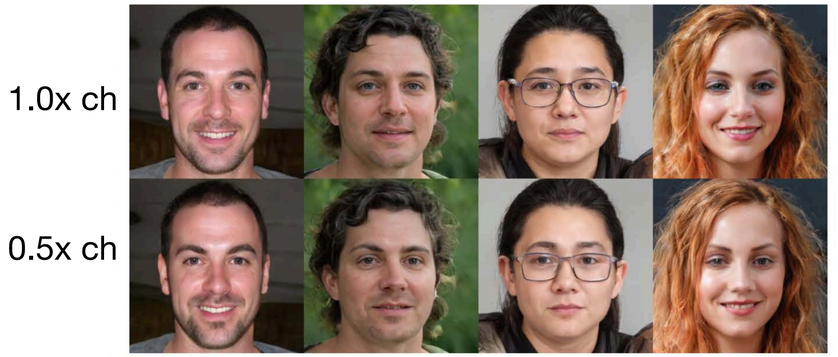
- **Anycost GAN**
  - Run a low-cost model to generate a fast, interactive preview
  - Run a full-cost model to generate the final, high-quality output (during idle time)
- Supports flexible **resolutions** and **channel widths**



Flexible resolutions

32x32 64x64 128x128 256x256

Flexible channel widths



Anycost GANs for Interactive Image Synthesis and Editing [CVPR 2021]

Anycost GAN

Tiered Pricing for Content Generation as a Service

- Good quality despite 5x computation reduction

MACs: 100% 1.0x reduction

Compute Budget	1x	0.7x	0.5x	0.4x	0.2x
Tiered Pricing	$0.01	$0.007	$0.005	$0.004	$0.02

The quality is still reasonably good

Anycost GANs for Interactive Image Synthesis and Editing [CVPR 2021]

Same Principle, Diverse Applications

Applications

please briefly explain large language model in one sentence.

A large language model is a type of artificial intelligence that can process and generate human-like language, based on vast amounts of data it has been trained on.

Large Language Model

Generative AI

Advanced Driver Assistance System

TinyML

Techniques

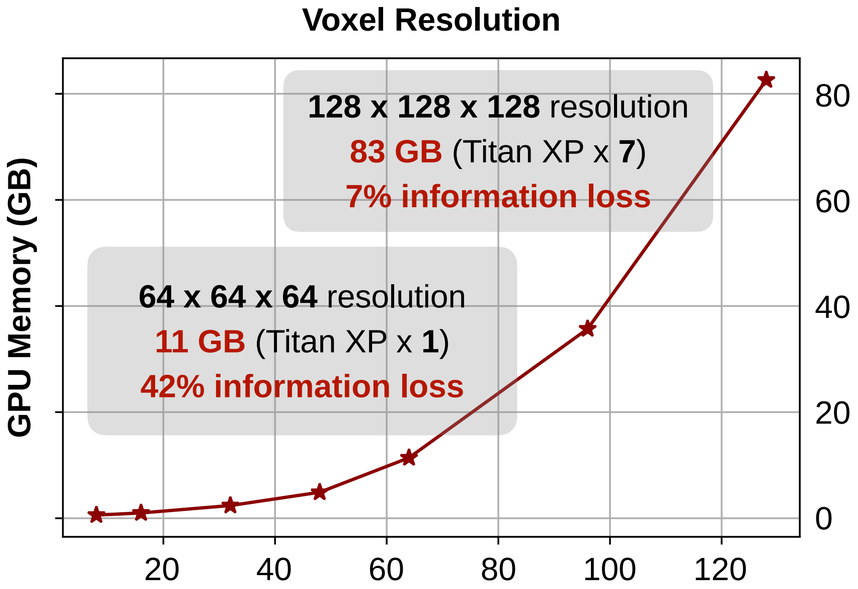
- Hardware-aware NAS
- Pruning & Sparsity
- Quantization
- Distillation
- New Primitive

PVCNN: Point-Voxel CNN

New primitive for handling spatial sparsity in point clouds

Voxel-Based Models

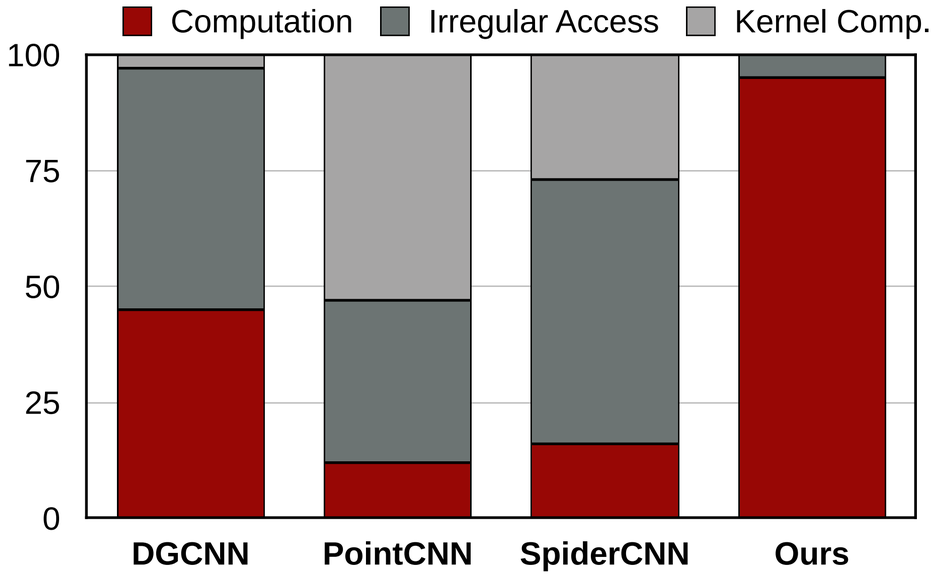
3D ShapeNets [CVPR'15]
VoxNet [IROS'15]
3D U-Net [MICCAI'16]



GPU memory consumption **increases cubically** with the volumetric resolution

Point-Based Models

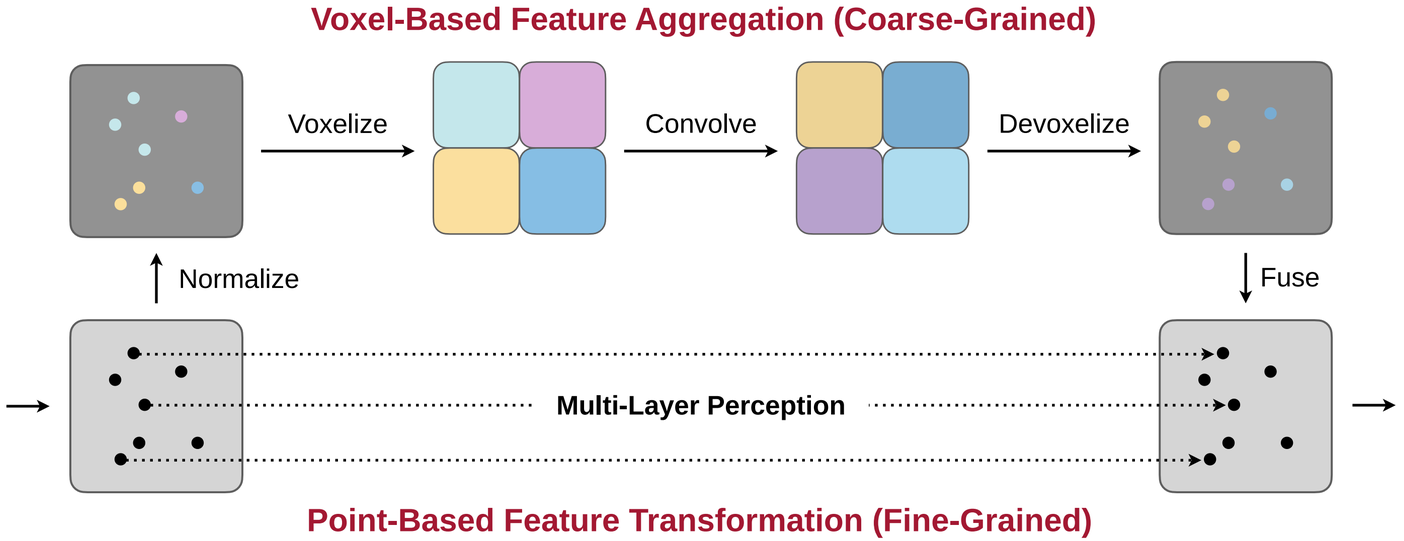
PointNet [CVPR'17]
PointCNN [NeurIPS'18]
DGCNN [SIGGRAPH'19]



Point-based models suffer from **random memory accesses** and **dynamic kernel computation**

PVCNN: Point-Voxel CNN

PVCNN is 2.7-6.9x faster than SOTA while being more accurate



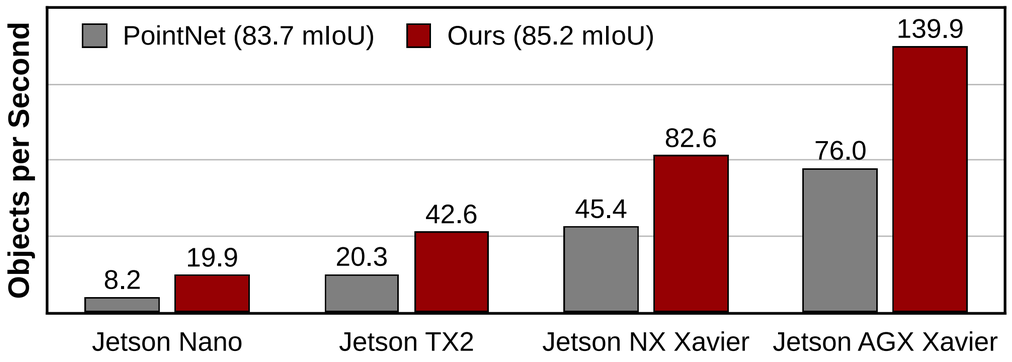
Comparisons with Point-Based Models

Voxel-based branch conducts convolution over a **regular grid representation**, which resolves the issue of random memory access.

Comparisons with Voxel-Based Models

Point-based branch captures **high-resolution** information **efficiently**, which resolves the issue of large memory footprint.

Real-Time Inference on Edge Devices

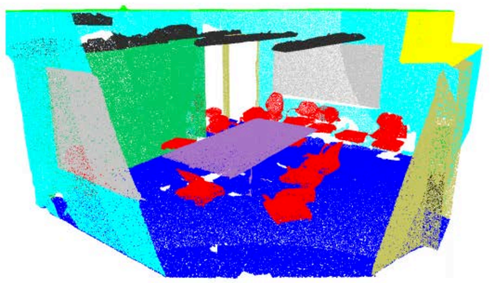


"**Project of the Month**" by NVIDIA Jetson Community

Object Part Segmentation

2.7x measured speedup
1.5x memory reduction

Indoor Scene Semantic Segmentation

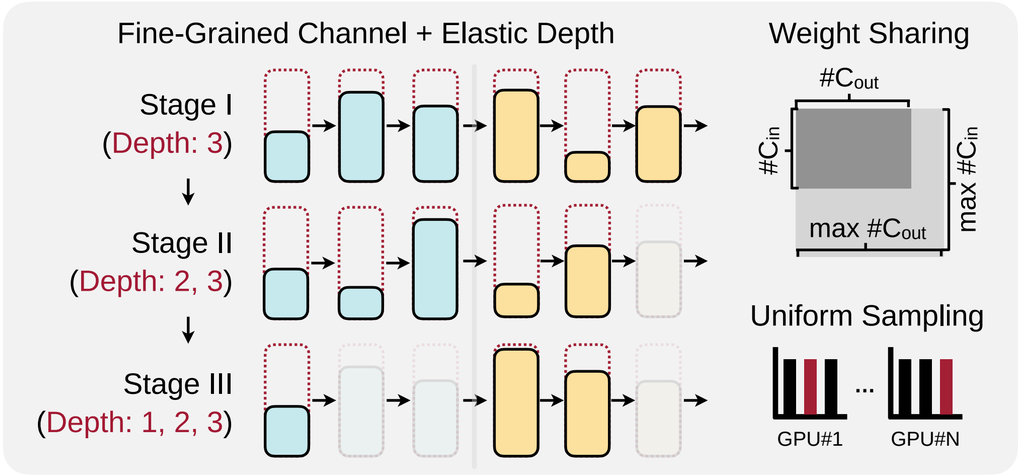


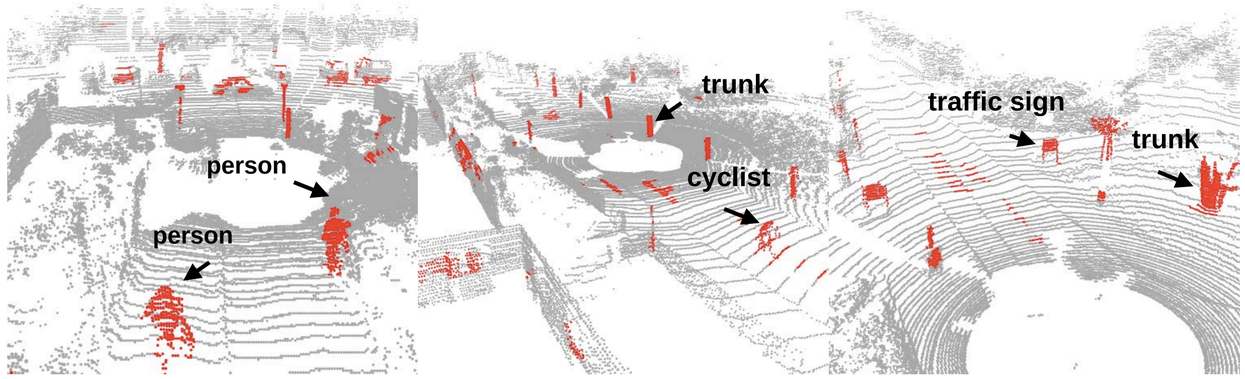
6.9x measured speedup
5.7x memory reduction

SPVNAS: 3D NAS for Sparse Point Clouds

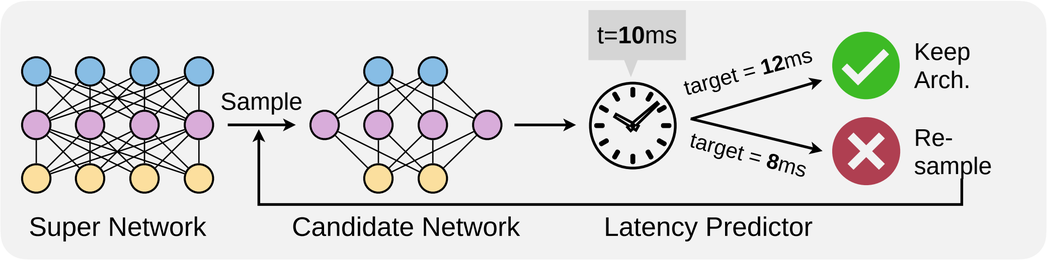
Neural Architecture Search for Point Cloud and LiDAR Perception

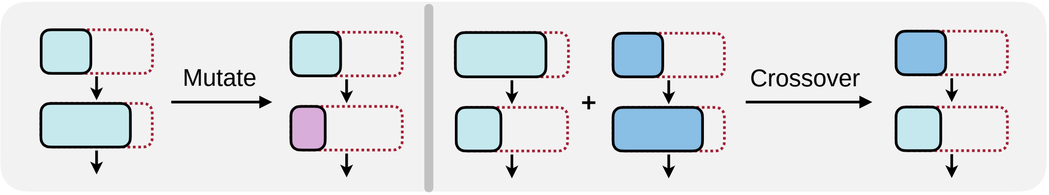
Super Network Training



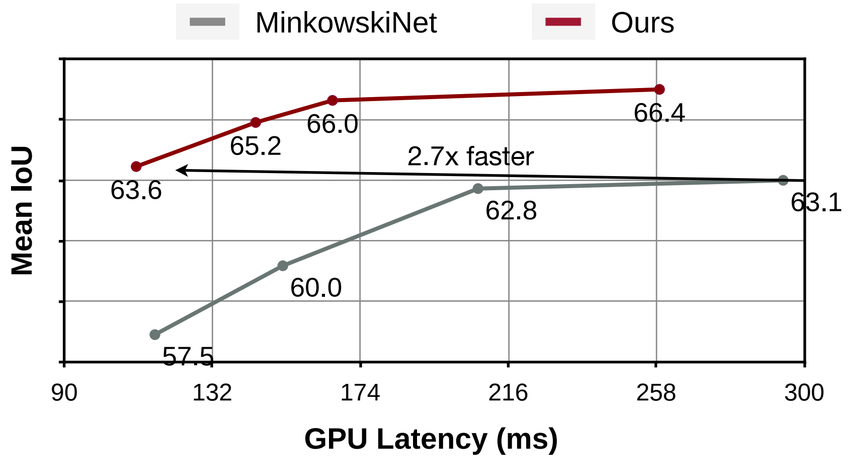


Evolutionary Architecture Search





Superior Efficiency-Accuracy Trade-Off (Especially Under Constrained Scenarios)



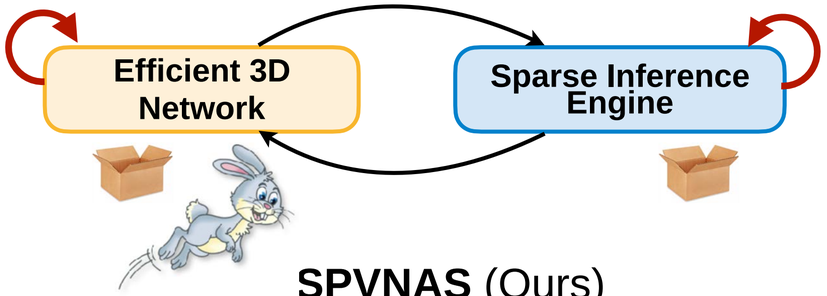
Difference from 2D NAS:

- **SPVConv** instead of 2D convolution as the building blocks;
- Enlarging design space on **channels** but shrinking search choices in **kernel sizes**.

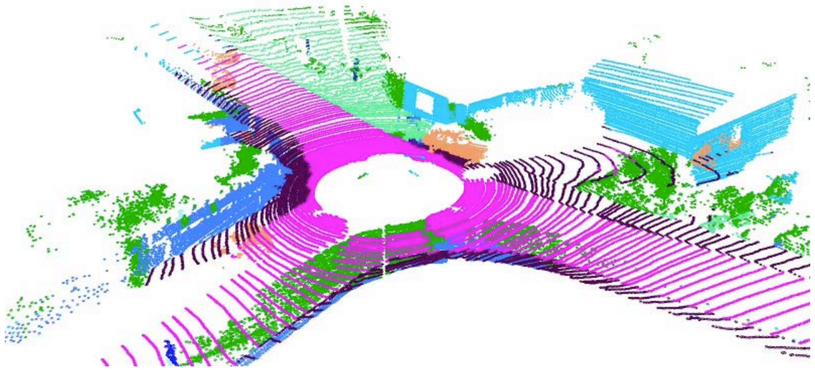
SPVNAS accelerated by TorchSparse

TorchSparse brings about another 1.3x speedup to the efficient SPVNAS model

System-Algorithm **Codesign**

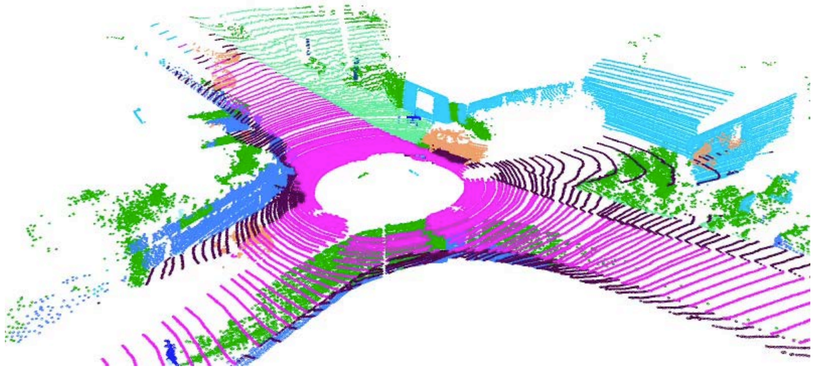


MinkowskiNet



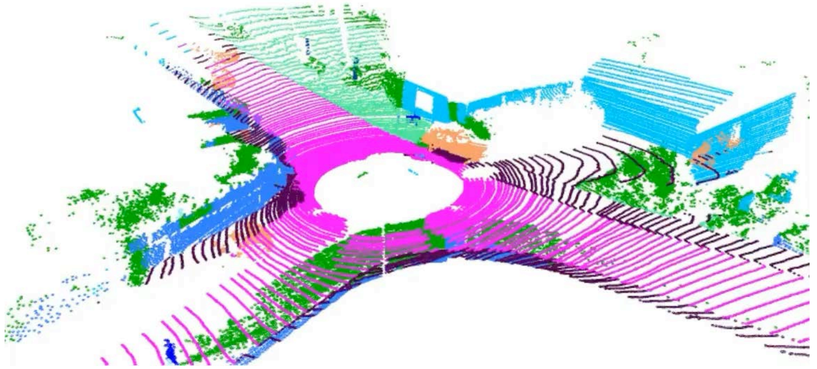
Mean IoU: **63.1** Throughput: **3.4** FPS
(**21.7M** Params **114.0G** FLOPs)

SPVNAS (Ours)



Mean IoU: **63.6** Throughput: **9.1** FPS
(**2.6M** Params **15.0G** FLOPs)

SPVNAS + TorchSparse (Ours)

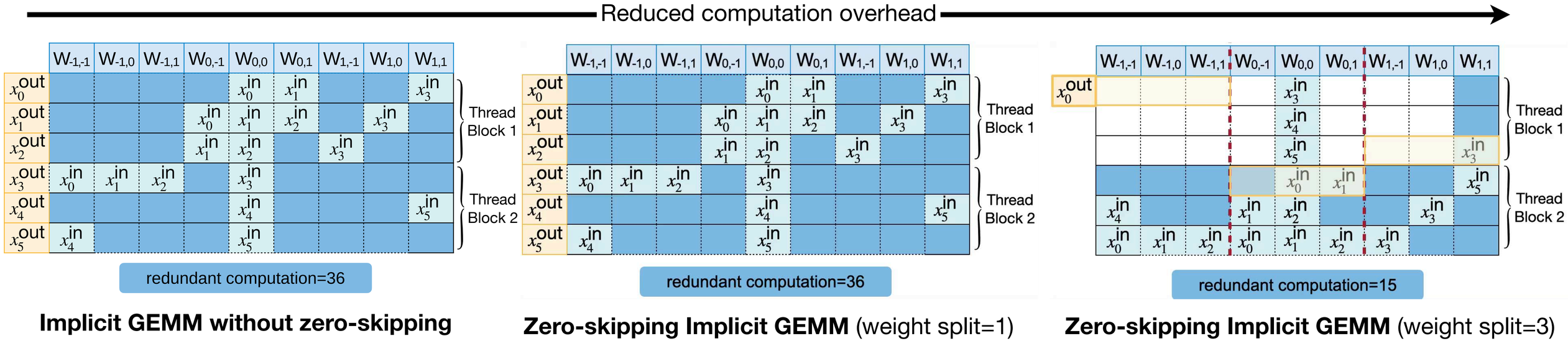


Mean IoU: **63.6** Throughput: **12.1** FPS
(**2.6M** Params **15.0G** FLOPs)

Measured on GTX1080Ti

TorchSparse: Efficient Point Cloud Library

Balancing computation regularity and computation overhead



Data Driven Auto-tuning based on profiling feedback

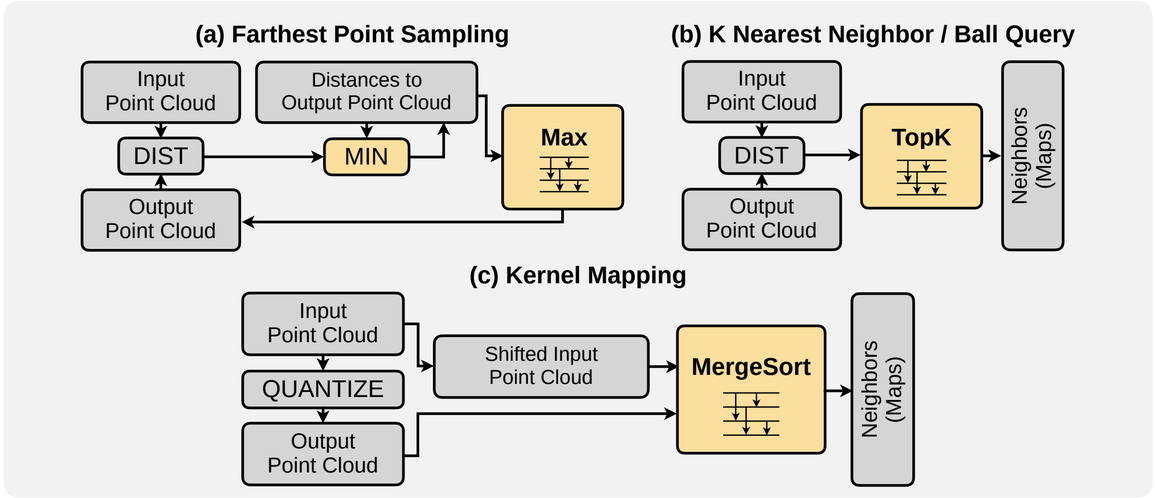
- Automatically determine the best tradeoff between **control flow** and **computation** overhead;
- **Mixed dataflow configurations** for different layers and forward/backward computation.

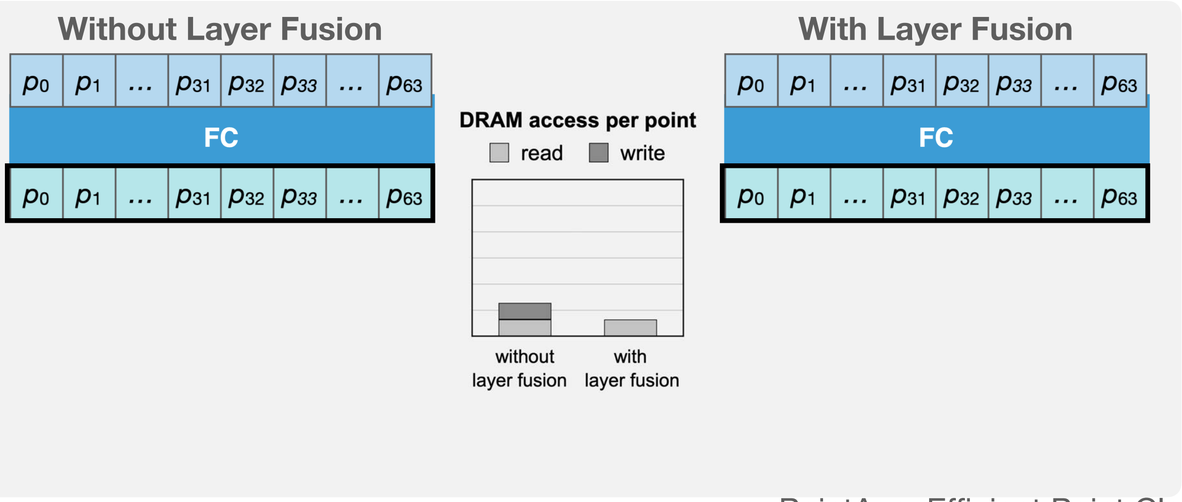
PointAcc: Point Cloud Accelerator

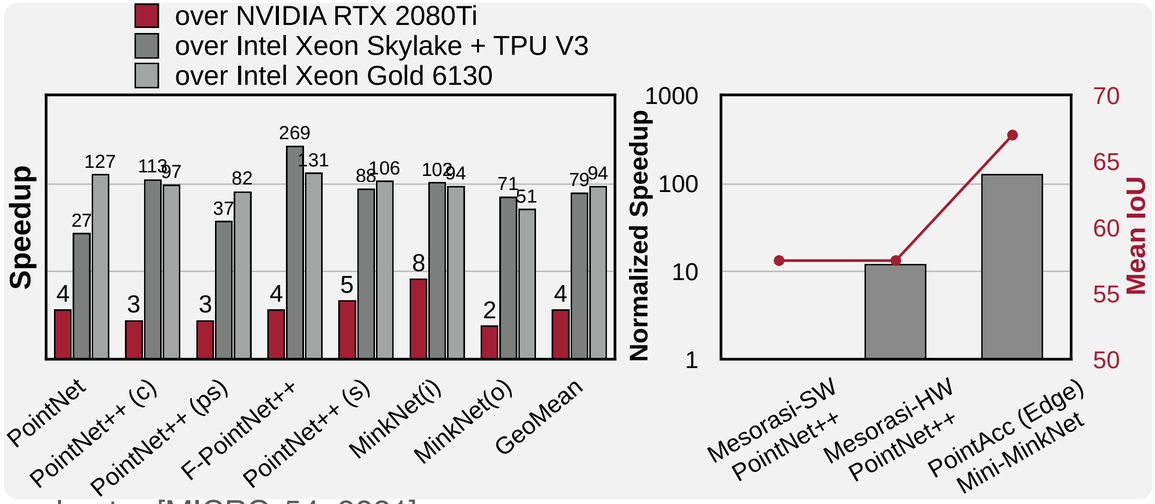
One hardware architecture, diverse neural architectures

- The sparsity of point clouds leads to two bottlenecks:
  - ***Diverse mapping operations*** for searching the input, output, weight maps (e.g., k-Nearest Neighbor, Ball Query, Kernel Mapping)
  - Data movement overhead from ***gather and scatter*** of the sparse features
- PointAcc maps diverse mapping ops into sort-based computation with ***one versatile hardware architecture***.
- PointAcc reduces off-chip memory access and minimize the overhead of gather and scatter by ***flexible caching*** and ***layer fusion***.

Diverse Mapping Ops in One Versatile Architecture



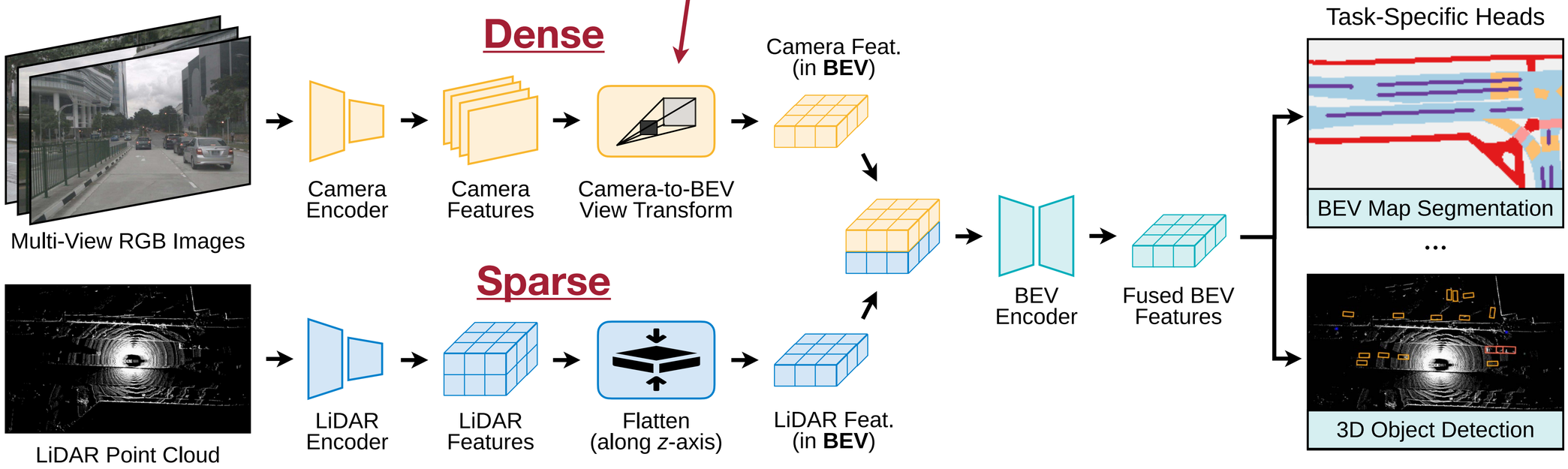


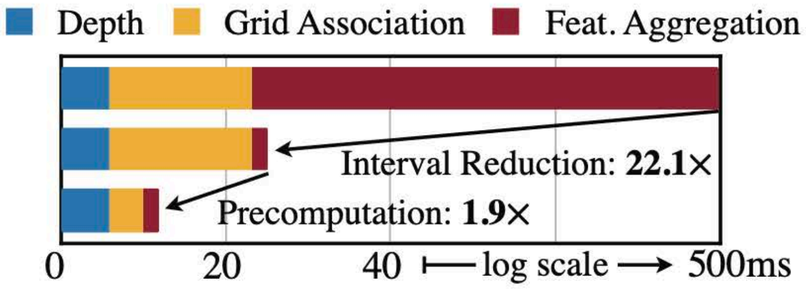


PointAcc: Efficient Point Cloud Accelerator [MICRO-54, 2021]

BEVFusion: Dense (Camera) + Sparse (LiDAR)

Accelerate LSS by 40x with Interval Reduction and Pre-computation





- Ranked **first** on **nuScenes 3D object detection** (2022/6).
- Ranked **first** on **nuScenes 3D object tracking** (2022/7).
- Ranked **first** on **Waymo 3D object detection** (2022/11).
- Ranked **first** on **Argoverse 3D object detection** (2023/4).

BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation [ICRA 2023]

BEVFusion: Multi-Task Multi-Sensor Fusion

BEVFusion takes **multi-modal** sensory inputs and supports **multiple** 3D perception tasks.

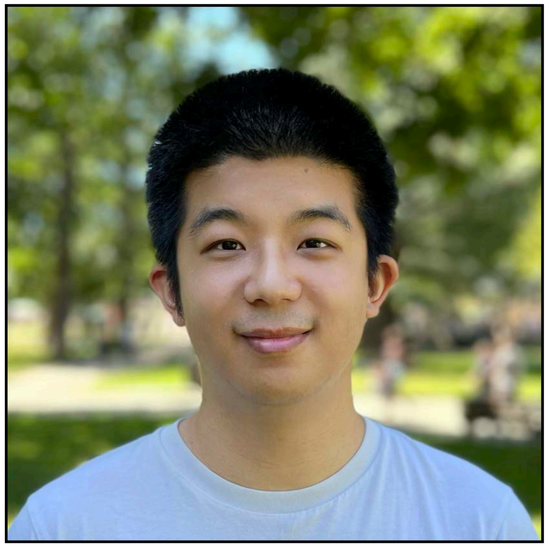
Multi-View Camera

LiDAR

https://youtu.be/uCAka90si9E

PhD Student — Zhijian Liu

Research Interest: **Efficient Algorithms and Systems for Deep Learning**
Graduating in Fall'23



Zhijian Liu

zhijian@mit.edu

https://zhijianliu.com

@zhijianliu96

Representative Work

- Algorithms:
  - **PVCNN** (NeurIPS'19 Spotlight)
  - **SPVNAS** (ECCV'20, TPAMI'21)
  - **FlatFormer** (CVPR'23)
- Systems:
  - **TorchSparse** (MLSys'22)
- Applications:
  - **BEVFusion** (ICRA'23)

Selected Honors

- Qualcomm Innovation Fellowship
- NVIDIA Graduate Fellowship
- MIT Ho-Ching and Han-Ching Fund Award

Same Principle, Diverse Applications

Applications

Techniques

Hardware-aware NAS

Pruning & Sparsity

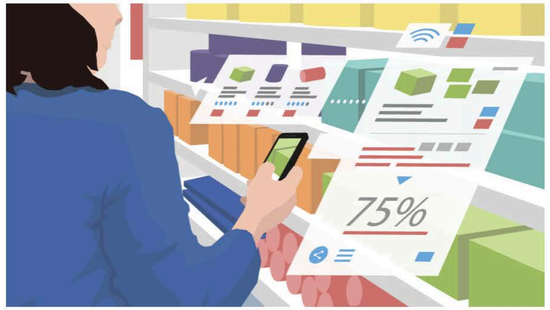
Quantization

Distillation

New Primitive

Background: The Era of AIoT on Microcontrollers

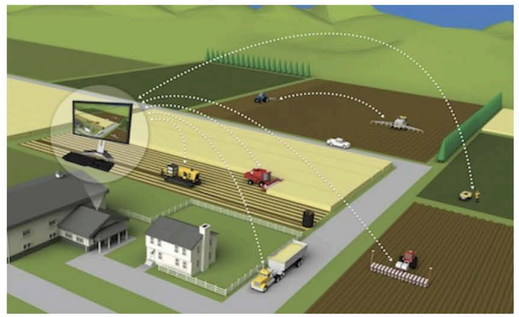
Smart Retail



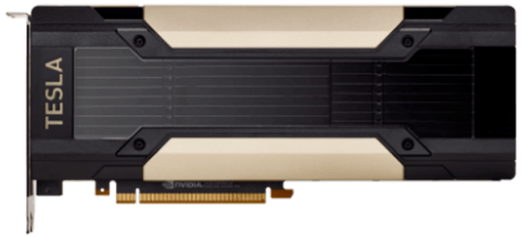
Personalized Healthcare

Smart Home

Precision Agriculture



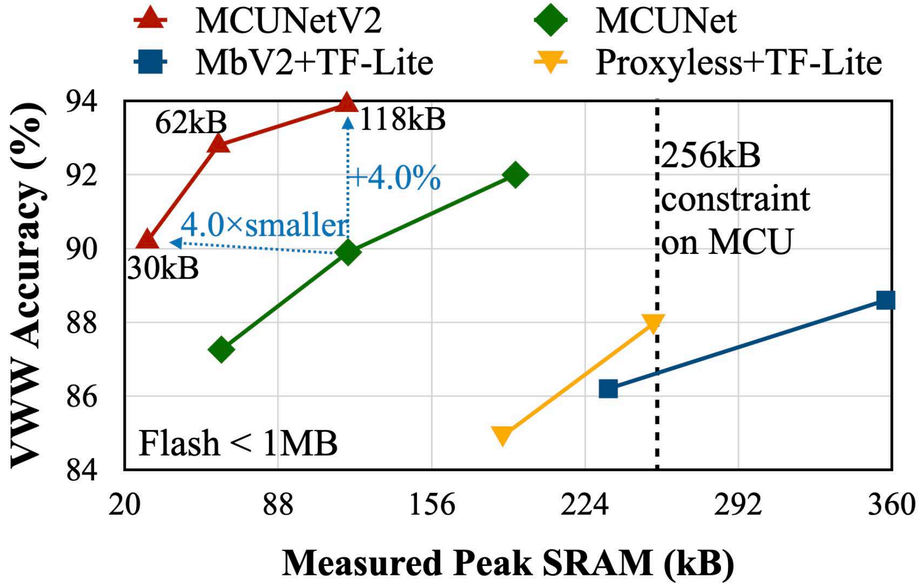
- **Problem**: restricted memory size



	Cloud AI	Mobile AI	Tiny AI
Memory (Activation)	32GB	4GB	320kB
Storage (Weights)	~TB/PB	256GB	1MB

MCUNet

Deploy AI on MCUs that has only 256KB SRAM



Face/mask detection

Person detection

The camera is OpenMV Cam.

Inference Is Good. Can We Learn on Edge?

AI systems need to continually adapt to new data collected from the sensors

Not only inference, but also training

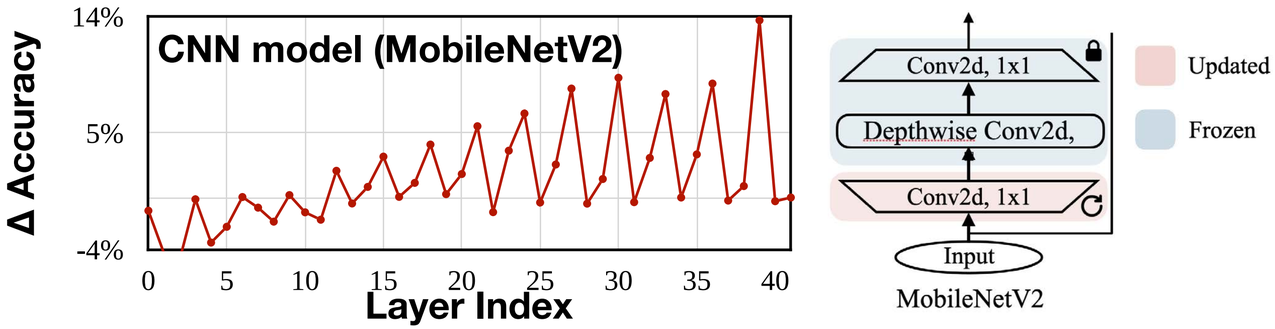
- On-device learning: **better privacy, lower cost, customization, life-long learning**
- Training is more **expensive** than inference, hard to fit edge hardware (limited memory)

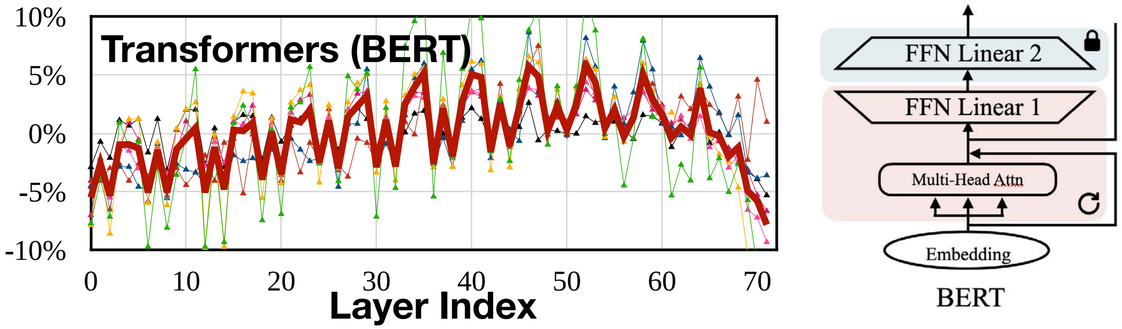
The camera is OpenMV Cam.

Sparse Training

Only update important layers and sub-tensors to save memory

- **Sensitivity analysis**

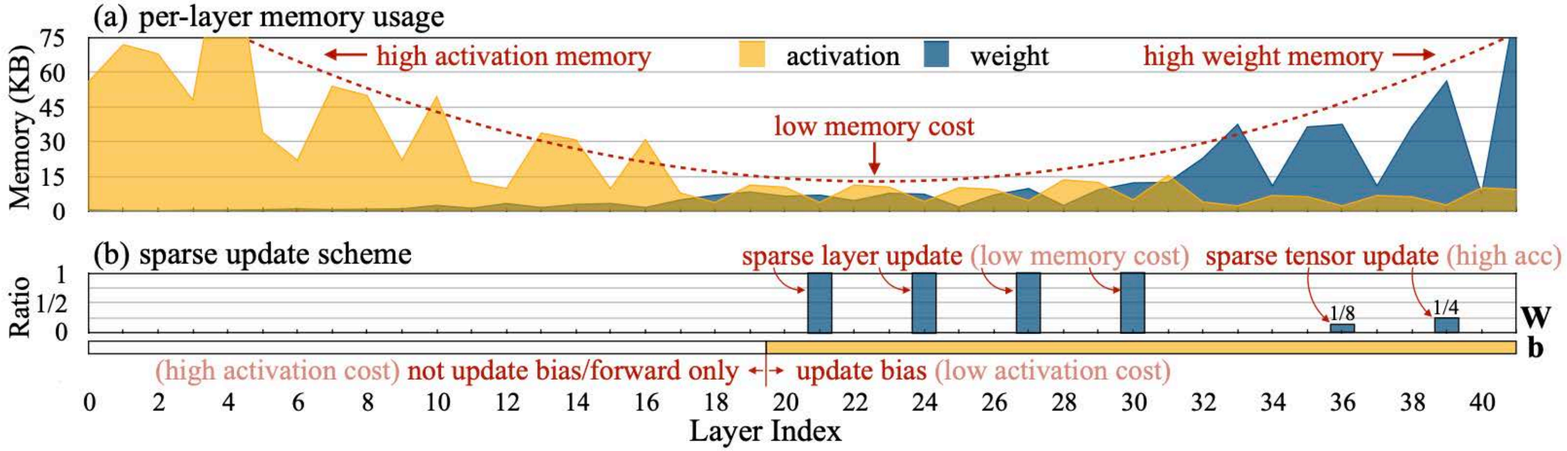




- Later layers are more important
- The first point-wise conv in each block contributes more

- Middle layers are more important
- Attention and first FFN layers contribute more.

- **Detailed Update Scheme for MobileNetV2**
  - The **activation cost** is high for the early layers;
  - The **weight cost** is high for the later layers;
  - The **overall memory** cost is low for the middle layers.
    - Bias-only update
    - Update weights for the middle layers

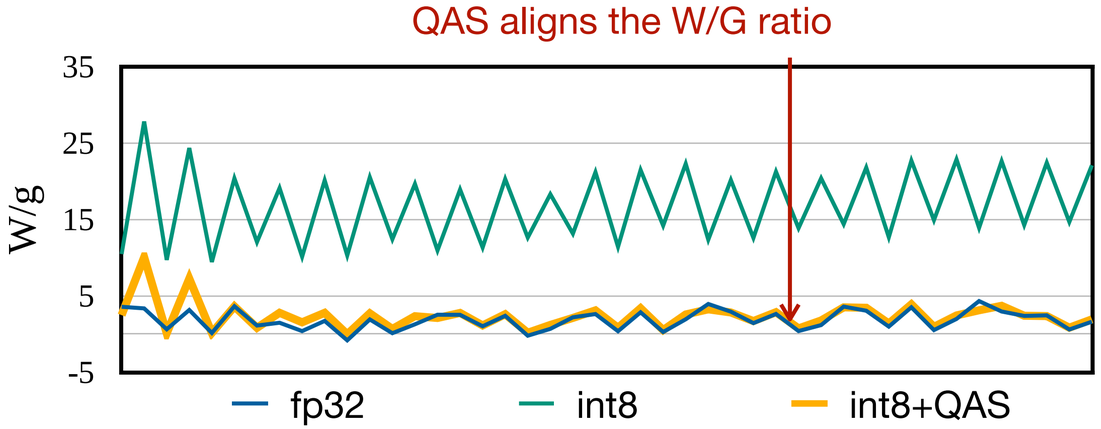


Low-Precision Training

with Quantization Aware Scaling (QAS)

- Optimizing an INT8 quantized graph leads to **memory and computing savings**
  - All weights and activations are in **INT8**
  - Different from quantization-aware training (QAT), where operations are performed in **FP16**

- … But at the cost of **worse convergence**
- We found the issue lie lies in gradient scale mismatch



- Solution: **quantization-aware scaling (QAS)**

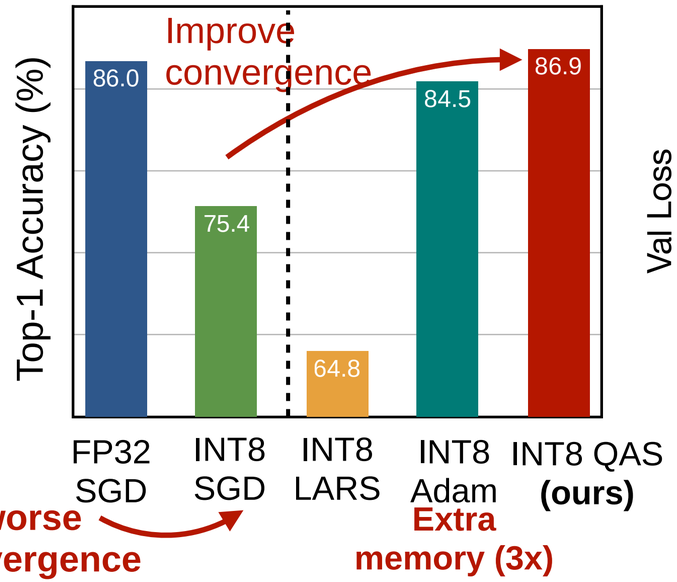
$$W = s_W \cdot (W/s_W) \overset{quantize}{\approx} s_W \cdot W_Q, \quad G_{W_Q} \approx s_W \cdot G_W$$

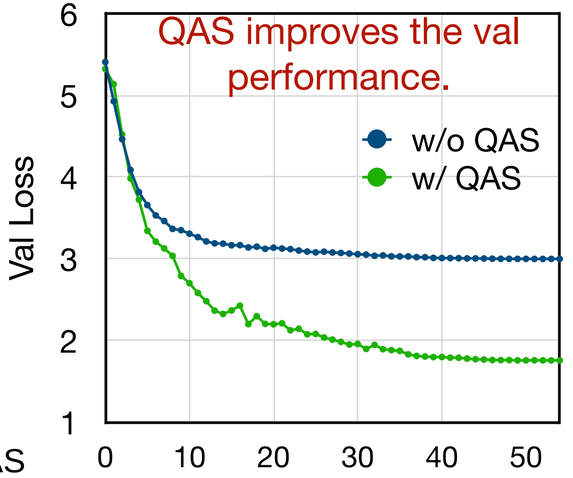
W_Q: [-128, 127], (W/s_W): [-2, 3]

weight and gradient ratios are off by s_W^{-2}

$$\|W_Q\|/\|G_{W_Q}\| \approx \|W/s_W\|/\|s_w \cdot G_W\| = s_W^{-2} \cdot \|W\|/\|G\|$$

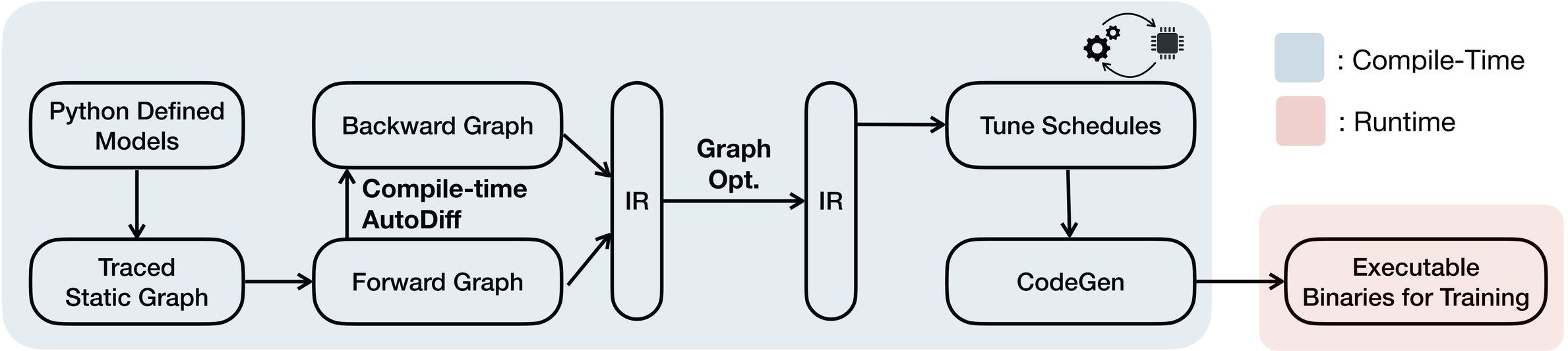
Thus, we need to re-scale the gradients $G'_{W_Q} = G_{W_Q} \cdot s_W^{-2}$

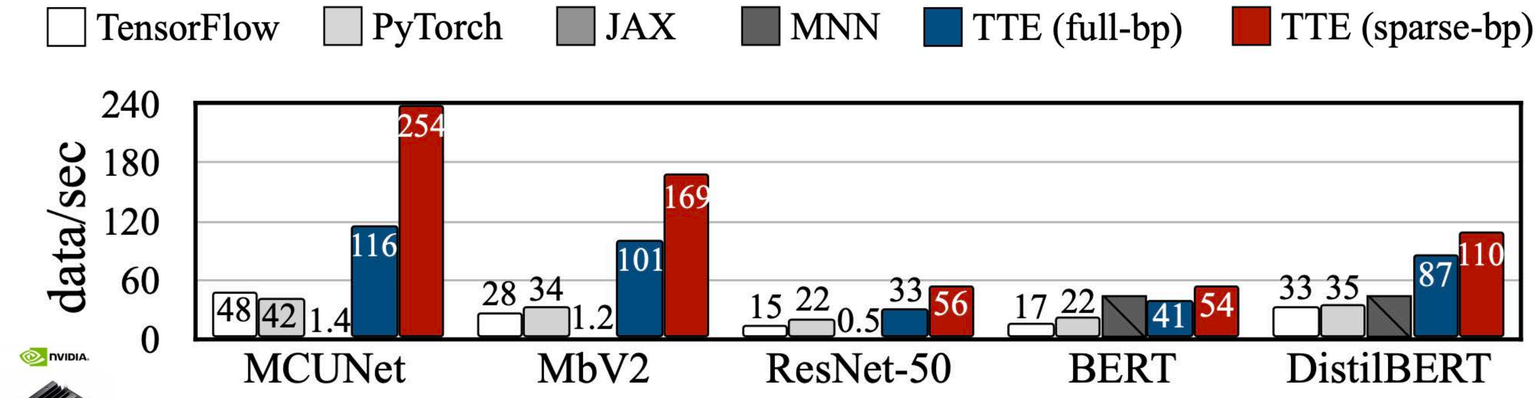




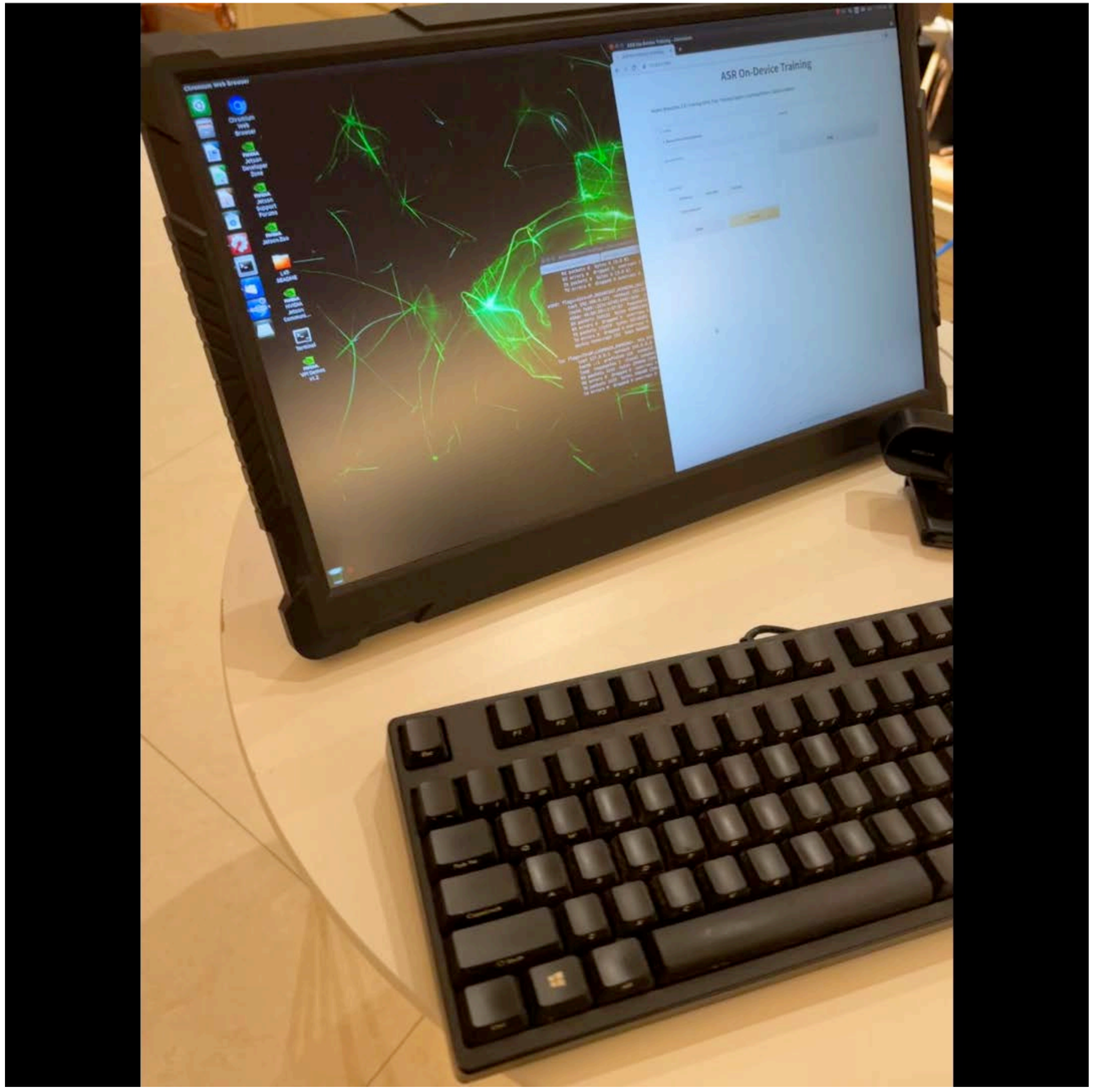
Tiny Training Engine

Translate the theoretical saving into measured savings. Runtime => Compile time



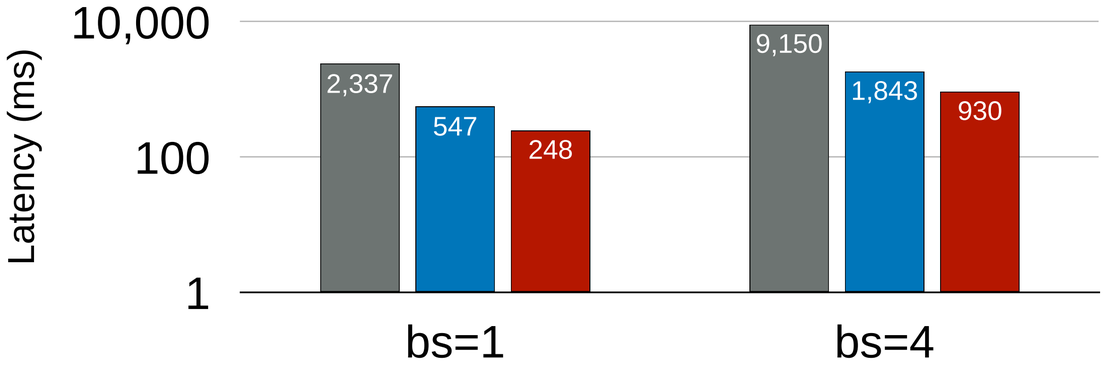


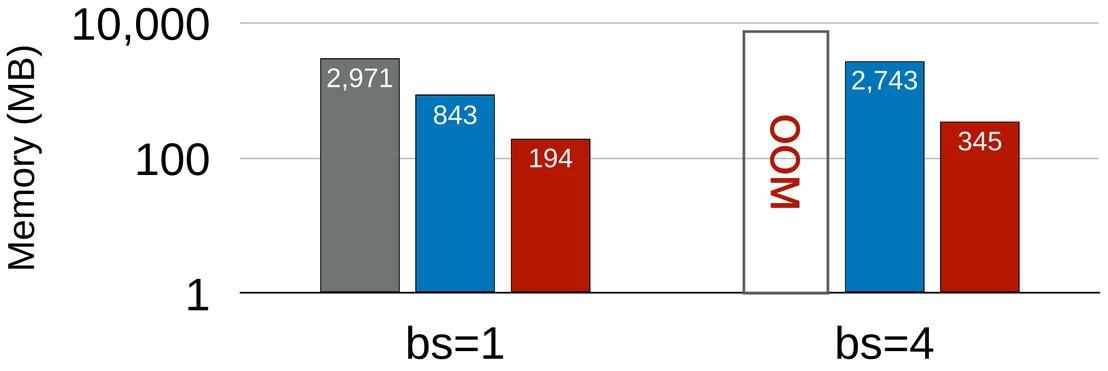
6x speedup compared to TensorFlow on Jetson Nano



PyTorch TTE (Dense) TTE (Sparse)

TTE On-Device Learning of Wave2Vec





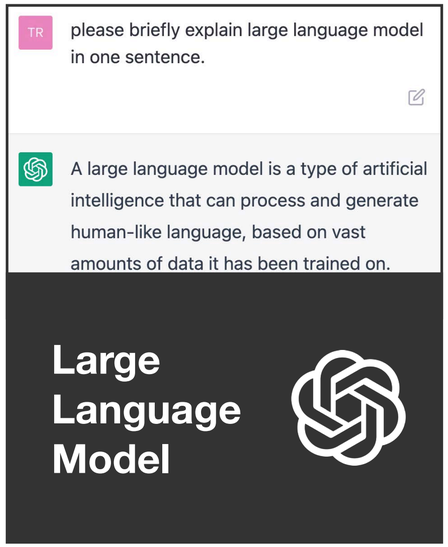
Accuracy (%)

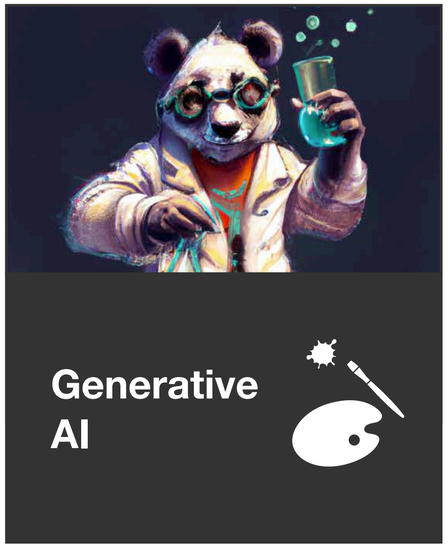
Wav2Vec2.0	Word Error Rate on TIMIT
Full update	33.9
Classifier-only update	98.7
Sparse Update (last two Encoder + FC)	33.3

Device: Jetson Nano; Backend:Tiny Training Engine; Task: Speech Recognition

Model Compression for Diverse Applications

Video Synthesis
Search Engine Revolution
Chatbots
Predictive Maintenance
Art Generation
Question Answering
Augmented Reality
Gesture Recognition
Storytelling
Autonomous Driving
Video Recognition
Music Composition
Sentiment Analysis
Blind Spot Detection
Health Monitoring
Fashion Design
Machine Translation
Adaptive Cruise Control

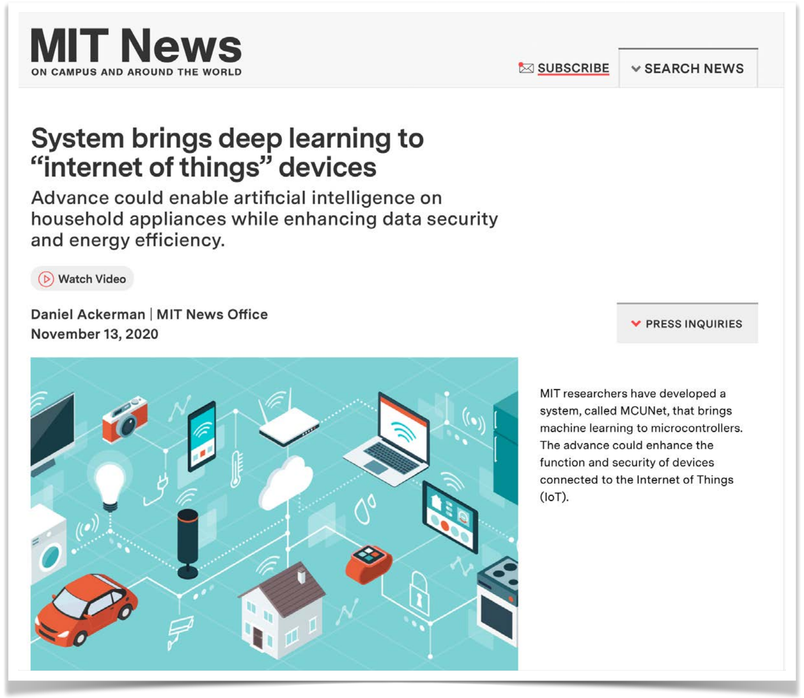




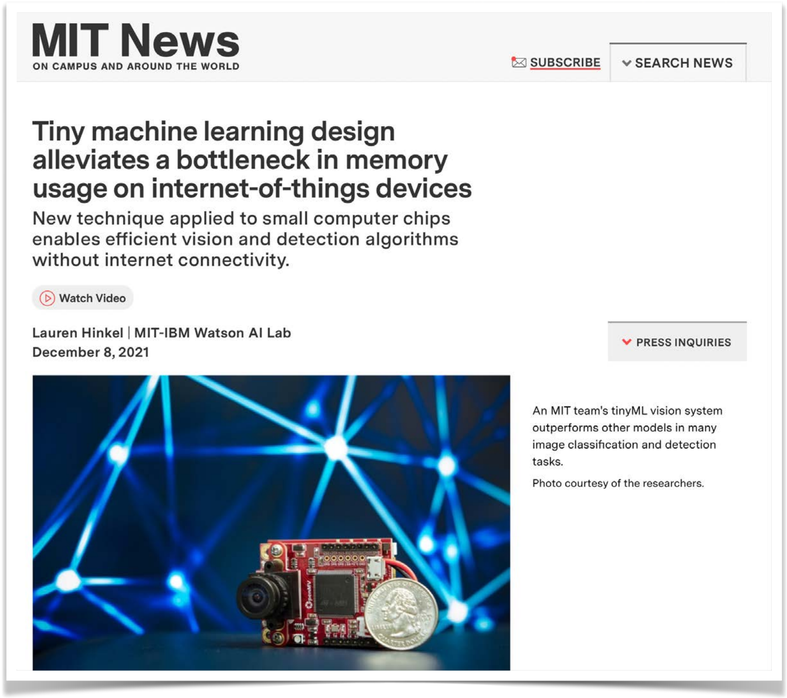
Application
(demand of computation)

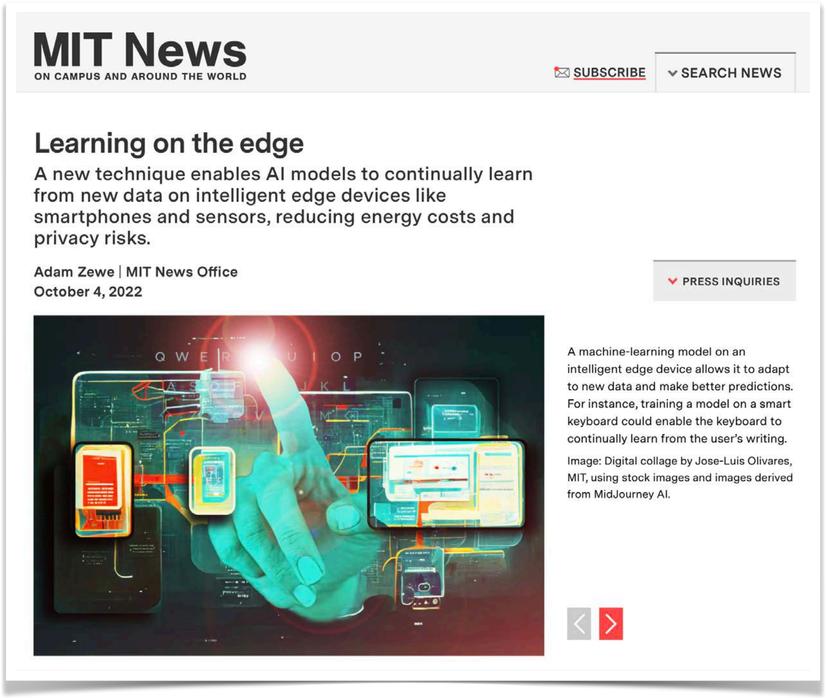
Hardware
(supply of computation)

Media



(Highlighted by MIT Homepage)





(Highlighted by MIT Homepage)

MCUNet: Tiny Deep Learning on IoT Devices [Lin *et al.*, NeurIPS 2020]
MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning [Lin *et al.*, NeurIPS 2021]
On-Device Training Under 256KB Memory [Lin *et al.*, NeurIPS 2022]

Open Source

Product Team Enterprise Explore Marketplace Pricing Search

mit-han-lab / mcunet Public

Code Issues 6 Pull requests Actions Projects Security Insights

master 2 branches 0 tags

README.md

MCUNet: Tiny Deep

This is the official implementation of th

website | paper | paper (v2) | de

Product Team Enterprise Explore Marketplace Pricing Search

mit-han-lab / tinyengine Public

Code Issues Pull requests Actions

master 1 branch 0 tags

README.md

TinyEngine

This is the official implementation of TinyEngine,
Microcontrollers. TinyEngine is a part of MCUNet,
co-design framework for tiny deep learning on mi
tight memory budgets.

The MCUNet and TinyNAS repo is here.

MCUNetV1 | MCUNetV2 | MCUNetV3

mit-han-lab / tiny-training Public

Edit Pins Unwatch 8 Fork 0 Star 65

Code Issues 1 Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

Lyken17 Merge branch 'main' of https://github.com/mit-han... f8dfb50 yesterday 4 commits

algorithm	prepare open source	2 days ago
compilation	prepare open source	2 days ago
figures	refine qas_accuracy figure	yesterday
.gitignore	prepare open source	2 days ago
.gitmodules	prepare open source	2 days ago
LICENSE	prepare open source	2 days ago
README.md	minor update	yesterday
assets	prepare open source	2 days ago
configs	prepare open source	2 days ago

README.md

On-Device Training Under 256KB Memory

About

On-Device Training Under 256KB Memory [NeurIPS'22]

tinytraining.mit.edu

edge-ai on-device-training learning-on-the-edge

Readme
MIT license
65 stars
8 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published

Sign up here to get updates!

https://forms.gle/UW1uUmnfk1k6UJPPA

New Course: TinyML and Efficient Deep Learning Computing

MIT 6.S965: https://efficientml.ai

6.S965 Logistics Schedule

TinyML and Efficient Deep Learning

6.S965 • Fall 2022 • MIT

Have you found it difficult to deploy neural networks on mobile devices and IoT devices? Have you ever found it too slow to train neural networks? This course is a deep dive into efficient machine learning techniques that enable powerful deep learning applications on resource-constrained devices. Topics cover efficient inference techniques, including model compression, pruning, quantization, neural architecture search, and distillation; and efficient training techniques, including gradient compression and on-device transfer learning; followed by application-specific model optimization techniques for videos, point cloud, and NLP; and efficient quantum machine learning. Students will get hands-on experience implementing deep learning applications on microcontrollers, mobile phones, and quantum machines with an open-ended design project related to mobile AI.

- **Time:** Tuesday/Thursday 3:30-5:00 pm Eastern Time
- **Location:** 36-156
- **Office Hour:** Thursday 5:00-6:00 pm Eastern Time, 38-344 Meeting Room
- **Discussion:** Piazza
- **Homework submission:** Canvas
- **Online lectures:** The lectures will be streamed on YouTube.
- **Resources:** MIT HAN Lab, Github, TinyML, MCUNet, OFA
- **Contact:** Students should ask all course-related questions on Piazza. For external inquiries, personal matters, or emergencies, you can email us at *6s965-fall2022-staff@mit.edu*.

Instructor Song Han
Email: songhan@mit.edu

TA Zhijian Liu
Email: zhijian@mit.edu

TA Yujun Lin
Email: yujunlin@mit.edu

Anonymous Student Feedback Collected from Mid-term

- This course is a deep dive into efficient machine learning techniques that enable powerful deep learning applications on resource-constrained devices.

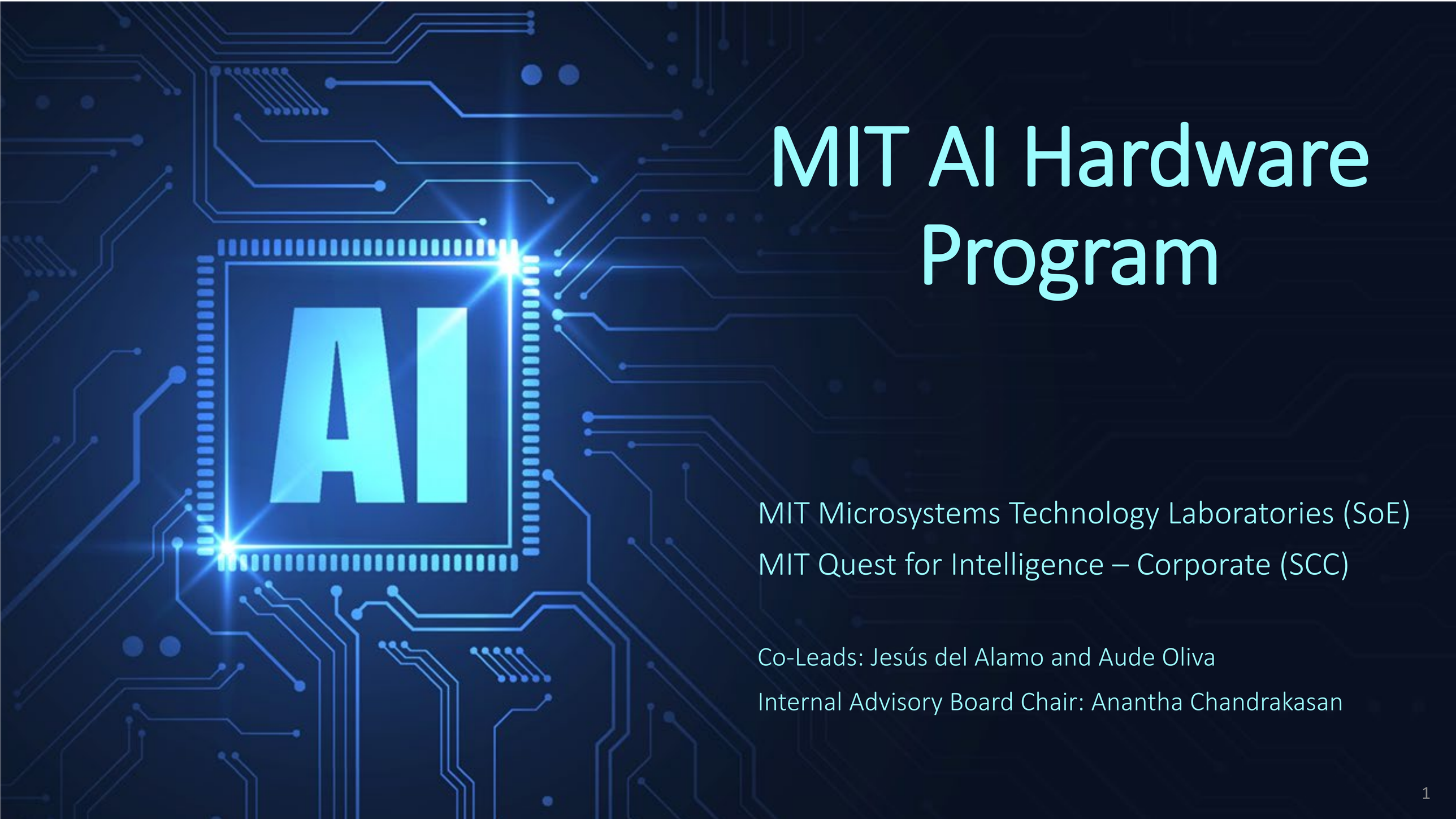
I really like how structured the labs are, and being able to see actual implementations of the techniques we learn about.

This is honestly one of the best set up courses I've taken at MIT

I love how we are using microntroller and focusing on application instead of just theories.

I managed the weekly labs and lectures by only watching the course on YouTube. As a researcher, I gained some valuable knowledge from your course. Excellent slides and teaching and useful labs.

I like the class and I have been able to follow the class easily (which had rarely happened to me in my previous courses)



MIT AI Hardware Program

MIT Microsystems Technology Laboratories (SoE)

MIT Quest for Intelligence – Corporate (SCC)

Co-Leads: Jesús del Alamo and Aude Oliva

Internal Advisory Board Chair: Anantha Chandrakasan

TinyML and Efficient AI

MIT HAN Lab
Hardware, AI and Neural-nets

github.com/mit-han-lab

youtube.com/c/MITHANLab

songhan.mit.edu
tinyml.mit.edu

Sponsors:

IBM, Qualcomm, intel, NVIDIA, XILINX, maxim integrated, NSF, amazon, Google, facebook, SONY, SAMSUNG, HYUNDAI, Ford

Media:

MIT Technology Review, IEEE SPECTRUM, WIRED, engadget, MIT News, AI DAILY, VentureBeat, ScienceDaily, Analytics Insight, AI Business